

国际信息工程先进技术译丛

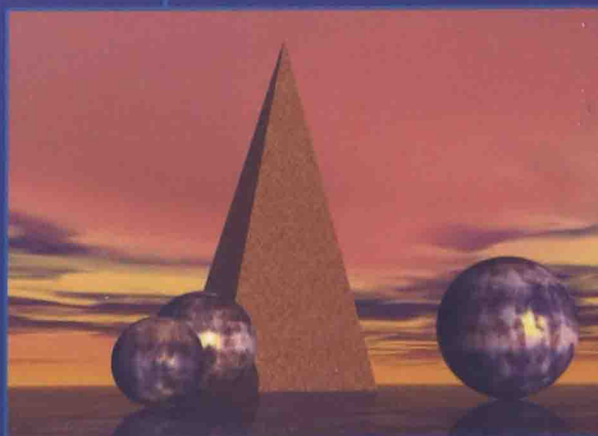


Springer

内容分发网络

Content Delivery Networks

Rajkumar Buyya
（澳） Mukaddim Pathan 编著
（希） Athena Vakali
宋伟 韩立伟 杨莉萍 郑睿 译



机械工业出版社
CHINA MACHINE PRESS



国际信息工程先进技术译丛

内容分发网络

（澳）Rajkumar Buyya
Mukaddim Pathan 编著
（希）Athena Vakali
宋伟 韩立伟 杨莉萍 郑睿 译



机械工业出版社

本书介绍了目前内容分发网络（CDN）最新的概念、原理、特点、应用、平台、设计思路、建模、仿真、工程方法以及最近的技术发展，可以帮助读者了解 CDN 的基本概念、技术原理及各种模型。

全书共分为 3 个部分。第 1 部分是 CDN 基础，分为 6 章，主要介绍 CDN 的基本思想、技术和现状。第 2 部分是 CDN 建模和性能，分为 4 章。第 3 部分是高级 CDN 平台和应用，分为 6 章。

本书内容详尽、章节顺序安排合理，既包括学术界及业界的研究成果，也不乏实际案例，可以作为系统工程师、相关从业人员、产品开发人员、研究人员以及研究生的参考书籍。

Translation from the English language edition

Content Delivery Networks/By Rajkumar Buyya, Mukaddim Pathan, Athena Vakali (eds.)

ISBN: 978-3-540-77886-8.

北京市版权局著作权合同登记图字 01-2009-3769 号。

图书在版编目（CIP）数据

内容分发网络/（澳）布亚（Buyya, R.）等编著；宋伟等译．—北京：机械工业出版社，2012.10

（国际信息工程先进技术译丛）

书名原文：Content Delivery Networks

ISBN 978-7-111-40052-3

Ⅰ．①内…　Ⅱ．①布…②宋…　Ⅲ．①计算机网络－网络结构　Ⅳ．①TP393.02

中国版本图书馆 CIP 数据核字（2012）第 243034 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）

策划编辑：张俊红　责任编辑：朱　林

版式设计：霍永明　责任校对：张　薇　闫玥红

封面设计：马精明　责任印制：乔　宇

唐山丰电印务有限公司印刷

2014 年 10 月第 1 版第 1 次印刷

169mm×239mm · 21.75 印张 · 467 千字

标准书号：ISBN 978-7-111-40052-3

定价：98.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务　　网络服务

社服务中心：（010）88361066　教材网：http://www.cmpedu.com

销售一部：（010）68326294　机工官网：http://www.cmpbook.com

销售二部：（010）88379649　机工官博：http://weibo.com/cmp1952

读者购书热线：（010）88379203　封面无防伪标均为盗版

译 者 序

随着信息技术的快速发展，互联网已成为人类信息社会的主要基础设施之一，但互联网经过半个多世纪的发展，已经演变成为一个复杂的巨系统。随着用户规模快速增长，资源并发访问显著，使得有限的网络资源与日益增长的用户需求之间的矛盾日益突出。此外，各种新型应用和智能终端的出现，种类繁多的业务形态访问异质异构资源时，服务质量要求差异较大，传统的内容传递方法难以满足要求。再者，网络流量视频化的趋势，需要通过新的网络体系架构解决海量数据的应用问题。这些均使得内容分发面临重大挑战。

内容分发网络（CDN）的思想产生于1998年，该技术通过内容复制和用户就近访问的方式，试图缓解困扰着互联网内容提供商的瓶颈难题。经过多年的实践化探索，CDN技术逐渐受到广泛关注并得到了快速的发展，其提供的分发服务已成为互联网中一项重要的组成部分，也有相关研究表明，通过对CDN的体系结构进行改进，性能进行优化，能够和现有的内容中心网络（CCN）的功能相媲美，即可以从CDN向未来网络进行演进。

同时，国外CDN的发展极为迅猛，已出现了如Akamai、Level3、Limelight等互联网知名大公司，且都有自己先进的产品和服务。国内也出现了蓝汛、世纪互联、网宿等相关企业，但市场占有量和国外大公司相比，还有一定差距。同时，为了更好地提供服务，原有的内容服务提供商，诸如腾讯等，也开始自建CDN。网络服务提供商也都想通过自有CDN占据网络流量的霸主地位。因此CDN竞争日趋激烈。

目前，国内在CDN技术方面的参考书籍较为匮乏，难以满足广大从业人员的需要。且本书包括CDN技术的概念、原理、性能分析、应用以及实验数据，凝聚着大量CDN研究者、开发人员、设计人员和行业人员的智慧和经验，章节脉络清晰、内容翔实，希望通过本书的引入，能够对从事该领域研究和开发的科研和工程技术人员带来帮助。基于此，我们组织了相关人员对本书进行翻译。

本书由中央民族大学的宋伟、中国兵器工业集团的韩立伟、中央民族大学的杨莉萍和郑睿进行翻译。翻译工作的分工如下：宋伟（1～3章，12～13章），韩立伟（4～7章），杨莉萍（8～11章），郑睿（14～16章），宋伟对全书各部分做了校订和修改。在本书的翻译过程中，力求既忠于原文，又适于国内从业人员阅读。由于译者知识水平和认识的局限，难免在内容和术语的翻译上存在不妥之处，真诚地欢迎专家和读者不吝指正，以完善我们的工作。同时感谢傅思遥在本书翻译过程中给予的帮助，并感谢出版社的张俊红、朱林等编辑在本书出版中付出的大量辛勤劳动。

本书的翻译受到“北京市教委共建项目 – 北京市公共安全信息监测平台关键技术研究”、“江苏省未来网络创新研究院 – 未来网络前瞻性研究项目”和“中央民族大学一流大学一流学科项目”的部分资助，这里一并表示感谢。

译　者

原书前言

作为一种无处不在的提供内容分享和服务的媒介，Web 网页的出现造就了互联网的快速发展。同时，访问 Web 内容和服务的用户数也呈指数增长，这就对提供内容和应用服务的 Web 系统以及互联网带宽提出了很高的要求，从而造成了许多网站无法满足这种要求，在提供及时的服务方面存在困难。

内容分发网络（CDN）的出现使这些问题得以解决，它以可扩展方式提供网络设施和运行机制，可以更高效地实现内容和服务的分发，增强了用户的网络体验。CDN 已经在许多领域得到了广泛应用，如学术机构、广告媒体和互联网广告公司、数据中心、互联网服务提供商（ISP）、在线音乐零售商、移动电话运营商、消费电子制造商以及其他的运营商等。随着 CDN 概念的扩展、成型和巩固，新形式的互联网内容和服务也随之不断涌现，为内容的分发和管理带来了新的挑战，同时也为 CDN 的构建、设计和实现带来了新的课题。因此，有必要对这个领域的技术发展趋势进行研究，从而为 CDN 的相关研究人员提供正确的研究路线图。

本书名为《内容分发网络》，将 CDN 最新的研究成果呈现给读者，包括概念、原理、特点、应用、平台、设计思路、建模、仿真、工程方法以及近期的技术发展。本书内容囊括了学术界及业界的研发成果，并对世界各地许多不同机构的实际案例进行了研究。同时，本书还为读者指出了未来有潜力的研究方向以及推动进一步创新的技术。作者希望本书能够成为一本有价值的参考书，为系统工程师、相关从业人员、产品开发人员、研究人员以及研究生等在内的广大读者带来帮助。

本书的概述和范围：本书将帮助读者理解 CDN 的基本概念，了解其基本技术，总结有关的概念、观点、原理以及应用于 CDN 领域中的诸多模型。因此，本书对内容的顺序进行了适当的编排，分别从以下几个方面对 CDN 进行了介绍，包括基本概念、设计过程、实践、技术、性能、平台、应用以及实验结果。同时，本书对与 CDN 相关的基本方法、创新性内容、重要研究成果以及进一步研究所用到的参考文献也都有所介绍，并在相应的章节对不同的设计和开发方法给出了比较，使新进入该领域的研究人员和资深研究人员都可以使用该评估结果作为研究的路线图。本书的所有内容都经过了审核、编辑和处理，并进行了适当的编排以保证其连贯性。这样，无论读者对 CDN 的了解和技术水平如何，都可以从本书中获得最大的收获。全书分为 3 部分：第 1 部分是 CDN 基础，第 2 部分是 CDN 建模和性能，第 3 部分是高级 CDN 平台和应用。这种内容安排保证了每个后续章节都是建立在之前章节的基础上，使内容之间实现了平滑的衔接。本书的主要内容如下：

- CDN 基础知识和现状；
- CDN 技术分类；
- 高效、动态和可扩展的内容复制技术；
- 内容分发和管理；
- 利用缓存进行复制的综合应用及其性能；
- 对动态内容请求的重定向；
- 利用经济学原理进行 CDN 建模；
- 定价策略与 CDN 商业模型；
- 资源分配与管理的数学建模；
- CDN 性能；
- CDN 互联方案、架构和方法；
- 流媒体；
- 动态 CDN 和基于 QoS 的自适应内容分发；
- 移动的动态 CDN；
- 应用：实时与按需视频服务、社区网络的内容分发。

第 1 部分主要介绍 CDN 的基本思想、技术和现状。在第 1 章，Pathan 等人介绍了 CDN 及其起源、发展和现状。该章定义了 CDN 及相关术语，对 CDN 进行了深入的介绍，并通过与相关的分布式计算模型的比较，总结了 CDN 的特点，进而提出了 CDN 领域未来技术的发展方向。目前已有大量文献对 CDN 的不同领域进行了研究，包括内容分发、复制、缓存以及 Web 服务器放置等。因此，在第 2 章 Pathan 和 Buyya 提出了一个综合性的 CDN 分类方法，覆盖了包括应用、特性和实现技术等在内的各个方面。在第 3 章，Chen 着重研究了 CDN 对高效、动态和可扩展复制技术的需求。针对该主题，作者介绍了在服务质量（QoS）和服务器的容量等约束条件下的动态和自组织的副本放置算法。在第 4 章，Cardelllini 等人主要研究了使用 CDN 进行内容分发的相关问题，其中重点介绍了对动态创建的个性化内容的分发。为了对各种缓存策略以及缓存和复制的综合应用进行分析和建模，Stamos 等人在第 5 章中介绍了相关的设计方法，并分享了实现经验，内容涉及在 CDN 仿真过程中的各种 Web 缓存问题。在第 6 章中，Ranjan 介绍了 CDN 中的请求重定向技术，同时也介绍了一种称为 WARD 技术的概念验证性实现，用来帮助实现动态内容的重定向。

本书的第 2 部分主要介绍 CDN 的经济学与数学建模及其性能。在博弈论的基础上，Christin 等人在第 7 章为参与 CDN 覆盖网的代理提出了一个基于成本的模型，并分析了在 CDN 中建立连接的激励因素。在第 8 章中，Hosanagar 讨论了内容分发市场的经济分析，提供了一个从 CDN 服务中捕获内容提供商收益的模型，并利用该模型讨论了定价策略。针对 CDN 领域中的资源管理和分配问题，Bektaş 和 Ouveysi 在第 9 章中展示了如何利用数学模型进行系统分析。Sitaraman 等人在第 10 章中介绍了 Akamai 的全局覆盖路由及其性能和可用性收益。

本书的第3部分主要介绍高级CDN平台及其极具吸引力的应用。Yoshida在第11章介绍了FCAN，这是一种动态CDN，主要用来缓解瞬时拥塞现象。Fortino等人在第12章介绍了一种基于CDN的支持协作播放服务的架构，并介绍了分级协作控制协议（HCOCOP），该协议主要用于在协作播放会话中的流媒体共享控制。在第13章，Czyrnek等人使用iTVP来解决多媒体CDN设计中面临的关键问题，iTVP是一种基于IP的、可以实现在国家级地域尺度上向大量在线用户分发多媒体内容的平台。在第14章，Loulloudes等人介绍了移动CDN的信息发布技术以及相关的挑战和现状。在第15章，Plagemann等人讨论了如何构建基于CDN的社区网络设施。在本书的最后一章，Pathan等人介绍了CDN之间的不同互联模型，并指出了这些模型在实现中面临的挑战。

致谢：本书得以完成，要感谢直接和间接参与其中的许多研究人员、学者、开发人员、设计人员以及业界人士。因此，我们对相关作者、研究机构以及本书所引论文、报告、文章、评论、网站以及学习材料的作者、研究机构和公司表示感谢。此外，本书的多位作者对其研究工作的支持机构和合作研究人员表示感谢，感谢他们对本书作者的研究所起到的重要影响。最后，我们要特别感谢Springer出版公司和出版编辑Christoph Baumann对我们的帮助，使得本书得以顺利出版。

上述的技术资源已在本书的适当位置通过引用表达了我们的致谢。对于本书中可能存在的不足之处，我们非常希望能够得到反馈意见，以便在下次修订时更正。

目　　录

第2部分　CDN 建模与性能

第 3 部分　先进 CDN 平台与应用

第1部分　CDN基础

第1章　内容分发网络：现状、观点和规律

Mukaddim Pathan，Rajkumar Buyya 和 Athena Vakali

1.1　引言

十几年来，网络用户见证了互联网的迅猛发展和成熟，宽带接入量的增长，系统复杂性的增加以及内容的日益丰富，使得网络流量呈指数级增长。这也给网络内容的优化管理和快速分发带来了新的挑战。例如，流行的 Web 服务由于大量请求而导致拥塞和瓶颈问题的发生。如美国 9·11 事件发生后，对该事件相关内容突发请求，产生了 flash crowd[14]，或 Slashdot 效应[11]，从而使相关服务器发生严重过载，导致“热点”产生[14]。这些突发性现象很可能会使该网站的服务器不堪重负，最终因无法承受流量的暴涨而导致服务器瘫痪，用户也会暂时无法获取网站的相关内容。

内容分发网络（CDN）[47, 51, 54, 61, 63]是一个新型的网络体系架构，通过若干互联网资源的相互协作，搭建互联网的覆盖网络，数据内容在若干镜像 Web 服务器上进行复制，目的是将内容以透明和高效的方式传递给终端用户。内容分发网络中分布式组件之间的合作在同构性节点和异构性节点上都可以进行。通过 CDN，用户访问 Web 内容时感知的服务质量（QoS）将得到解决。CDN 通过最大化带宽使用率、提高成功访问率、维护正确无误的内容复制，从而提升服务质量。下面将简要介绍内容分发网络的基本功能：

（1）请求重定向和内容分发服务

将用户请求重定向到最邻近和最合适的 CDN 缓存服务器处，从而绕过堵塞的网络节点，解决 flash crowd 和 Slashdot 效应。

（2）内容外包与分发服务

将内容从源服务器复制和（或）缓存到分布式的 Web 服务器。

（3）内容协商服务

满足不同个体用户或群体的特定需求。

（4）管理服务

通过管理不同的网络组件来实现记账，并监测和报告内容的使用情况。

CDN 主要的应用领域有公共网络、企业网络和边缘服务。由于 CDN 是一个快速

发展的研究领域，不断出现新的研究和创新，这就使本章无法对该技术进行面面俱到的介绍。因此，本章仅仅是对 CDN 的发展现状做一个概述。但有关 CDN 核心技术和原理的介绍，将对读者理解 CDN 技术起到积极和重要的作用。

本章余下内容组织如下：首先简要介绍内容分发网络，随后描述 CDN 发展的背景，并通过与其他分发式计算方法之间的比较来突出 CDN 的独特性。在 1.4 节，介绍了 CDN 的商业技术指标，1.5 节描述了 CDN 的发展现状，1.7 节会提出未来内容分发网技术的发展方向及研究路线图，1.8 节进行全章的总结。

1.2 概览

图 1.1 显示了一个 CDN 的模型。其中 CDN 的边缘分布着拓展到全球的 Web 服务器集群，端用户与集群相连。CDN 将内容传输给这些 Web 服务器集群，并进行全球范围上的发布，从而将内容快速、有效和可靠地传递给端用户。内容的复制将内容推送到分布式 Web 服务器，既可以是按照用户的请求需求完成的，也可以是预先复制的。因此，用户在不需了解这些细节的情况下，通过与其最近的副本 CDN 服务器进行通信，获取相关数据。

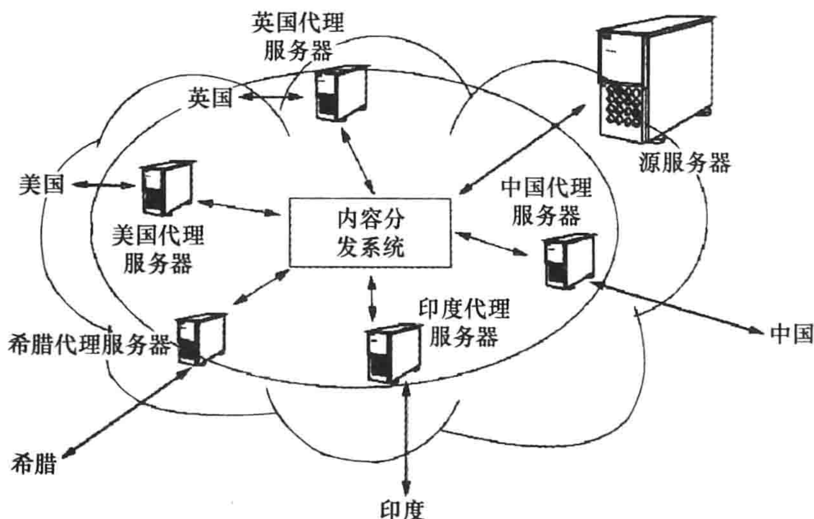


图 1.1　一个 CDN 模型

1.2.1 术语解释

在 CDN 的语境下，内容分发描述的是一种为满足终端用户请求而做出的内容服务行为。这里的内容可以是指任何数据资源，包括两大部分：编码媒体数据和元数据[53]。编码媒体包括静态、动态和流媒体数据（如音频、视频、文档、图片和网页）。元数据是对内容的描述，用于多媒体数据的标识、发现以及管理内容的统称，并有助于对多媒体数据的解释。内容既可以预先录制，也可以在直播过程中提取，既可以是长期保存的数据，也可以是系统短期存放的数据[53]。CDN 可以被看做是开放

系统互联（OSI）网络参考模型[32]的一种新型虚拟覆盖网络，它提供了基于应用层协议（如超文本传输协议（HTTP）和供实时传输流协议（RTSP））的覆盖网络服务[26]。

CDN 系统包括三个实体：内容提供商、CDN 提供商和终端用户。内容提供商或客户（customer）是拥有相关 Web 的 URL 且需要进行相关内容分发的实体，其源服务器上保存着大量需要分发的内容。CDN 提供商是一个专有的组织或公司，它们向内容提供商提供网络基础设施，以达到内容的有效实时传输。终端用户或客户端（client）是通过各个内容提供商的站点访问内容的用户。

CDN 提供商通过位于不同地域的缓存和（或）副本服务器来复制内容。CDN 缓存服务器通常也被称为边缘服务器或代理缓存服务器（surrogate）。CDN 的边缘服务器被统称为一个 Web 集群。CDN 通过让不同的边缘服务器共享的内容和 URL 来发布内容。用户的请求被重定向到与其距离最近的边缘服务器，并由该服务器按需将相关内容发送至端用户，此过程对端用户是透明的。此外，边缘服务器将所分发内容的记账信息传输到 CDN 的记账系统，以达到流量报告和计费的目的。

1.2.2 CDN 组件

图 1.2 展示了 CDN 系统的总体架构，主要包含四大组件。

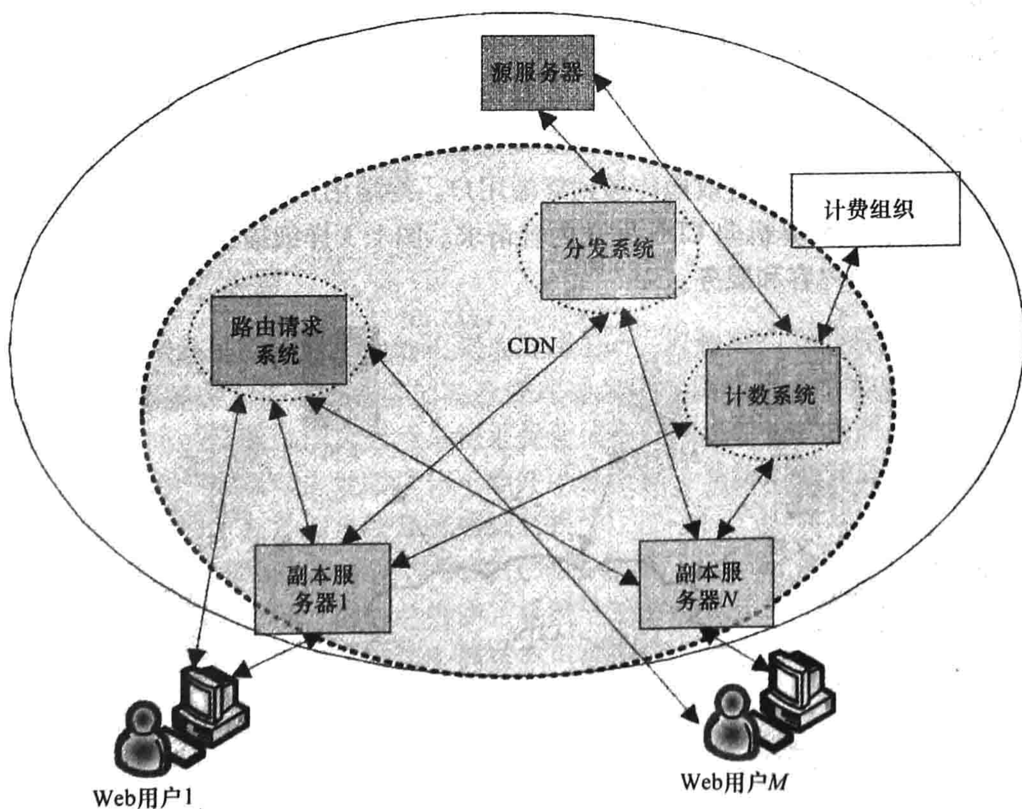


图 1.2　一个 CDN 的架构组件

(1) 内容分发组件

内容分发组件包括源服务器和一系列副本服务器，通过它们把内容的副本分发给端用户。

(2) 路由请求组件

路由请求组件负责将用户的请求分发给相应的边缘服务器，并与内容分发组件交互，以及时更新 CDN 缓存中存储的内容。

(3) 发布组件

发布组件把内容从源服务器传送到 CDN 边缘服务器，并保证缓存中内容的一致性。

(4) 记账组件

记账组件保存用户的注册记录和 CDN 服务器的使用记录。这些信息可以被内容提供商自身或第三方的收费组织用于流量报告和与使用量挂钩的收费系统。

CDN 的重点在于搭建自身的网络基础设施，以提供如下服务和功能：存储和管理内容、由边缘服务器分发内容、缓存管理以及静态、动态和流媒体内容的分发、数据备份、监控、性能测量以及报告。

内容提供商（用户）可以与 CDN 提供商签订服务协议，在 CDN 缓存服务器上存放内容提供商的内容。实际上，CDN 通常会对以下第三方内容进行托管：静态内容（如 HTML 页面、图像、文档和软件包等）、流媒体（如音频、实时视频）、用户上传视频（UGV）和其他内容的服务（如目录服务、电子商务及文档传输服务）等。内容的来源包括大型企业、网络服务提供商、媒体公司、新闻广播等。CDN 的典型用户是媒体和网络广告公司、数据中心、互联网服务提供商（ISP）、在线音乐零售商、移动运营商、消费类电子产品制造商以及其他运营公司等。这些用户都希望能通过网络将需要发布的内容快速有效地传递到终端用户。终端用户可以通过手机、智能电话/PDA、手提和台式计算机向 CDN 发送内容请求。图 1.3 详细描绘了 CDN 提供商向终端用户提供的不同内容和服务。

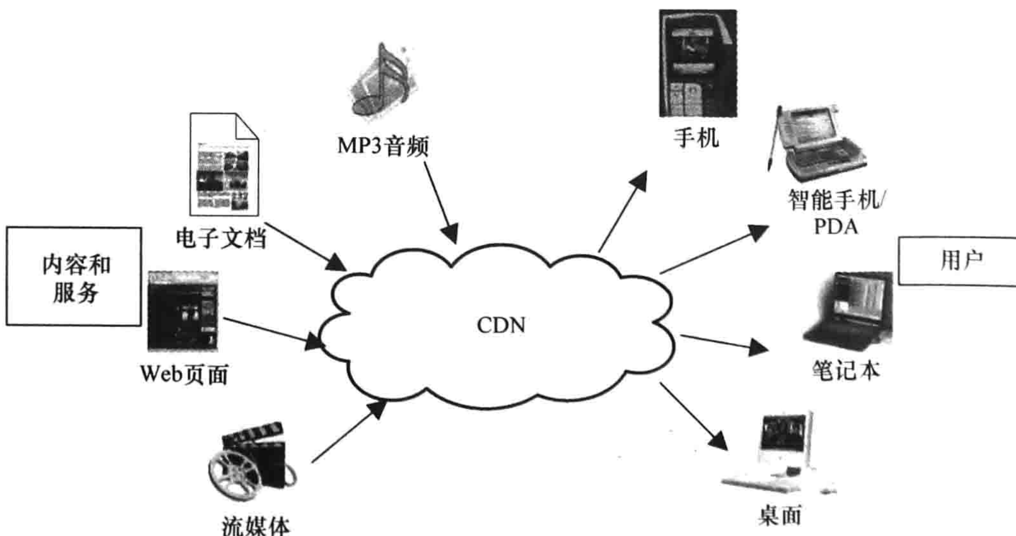


图 1.3 一个 CDN 提供的内容/服务

CDN 提供商根据终端服务器分发的内容（如流量）来收取客户的费用。CDN 具备一个专门的记账系统，可以收集和记录用户在路由请求、内容分发和传输方面的使用信息[26]。这个系统实时记录 CDN 各个组件的相关信息，用于统计、收费以及维护等用途。CDN 服务的平均收费标准是很高的[35]，通常超出了一般中小型企业和非盈利组织的承受能力（CDN 定价机制在走向平民化）。CDN 服务的价格主要是由以下几个最重要的因素[47]决定：

1）带宽使用，根据内容提供商统计的带宽使用量（每兆比特），按月向用户收取。

2）流量波动，在一些网络阻塞或不稳定状态（如突发性的流量）情况下，流量的定价会有所不同。

3）边缘服务器上复制内容的大小，（如每 GB 的单价）是一个重要的定价衡量标准。

4）边缘服务器的数量，为了使收费不高于通常意义下的缓存服务方式，CDN 提供商在提供内容服务时的一个重要定价因素。

5）整个 CDN 系统的可靠性与稳定性，以及内容分发外包的安全性，也制约了使用共享类保密数据的价格，这种数据的服务价格根据保护内容的类型和内容提供商而不同。

1.3　背景和相关系统

内容提供商常把 Web 作为一个向用户提供传输丰富多媒体内容的重要方式。但是，服务质量的下降以及长时间的接入延时（主要是由于下载时间过长引起），会让用户放弃使用而离开。一个企业通过网络在电子商务上获利越多，就越希望能通过改善上网用户的体验来增加自身网站的浏览量。因此，近几年相关技术的发展一直在力图提高内容分发的效率和服务质量，于是，一种支持该技术的新型网络架构——内容分发网络应运而生。

1.3.1　CDN 的发展

许多内容网络试图通过不同的机制来提高 QoS，以解决性能的瓶颈。

1）早期的解决方法是借助增加高速处理器来优化网络服务器的硬件水平，并通过提高存储空间或增加多任务处理系统来优化传统的网络结构。然而，这种方法的灵活性不佳，因为一些细微之处的优化可能并不会对整个系统的性能提升起到很大的作用，有时候甚至整个服务器系统都需要被替换[31]。

2）通过 ISP 来部署具有缓存功能的代理服务器（caching proxy），该方法对窄带用户访问网络会很有帮助，因为它提高了系统性能并降低了网络使用率，使缓存代理服务器更加贴近用户。缓存代理服务器还有可能配置一些可以探测到服务器失效的技术，从而使缓存资源的有效性最大化。用户通常会设置他们的浏览器，通过向缓存，而不是直接向源服务器来发送 Web 请求。当设置成功后，用户的整体浏览过程通过缓存代理服务器提供服务，这些缓存代理服务器上存储着用户访问量最高的内容。

3）提供商还可能会为不同地理分布的地区部署不同级别的缓存（本地级、地区级和国际级）。这种方式属于分层缓存方法，可以进一步改进性能和带宽存储能力[17]。另一种更广泛使用的方法是建立可扩展的服务器群（server farm）。一个服务

器群由多个 Web 服务器组成，每一个服务器都负责承担相同 Web 站点的响应负载[31]。这使得 4～7 层交换机（一种依据 URL 请求、内容类型和用户名等信息工作的智能交换，而这些信息都可以在 4～7 层 OSI 协议栈请求包里找到）、网页交换机或内容交换机可以检查被请求的内容，并在各组服务器之间调度。服务器群由代理服务器而非交换机构成[24]，这种方式更加灵活，可拓展性也更强。此外，它还有内部纠错能力。随着网站与互联网之间网络连接的升级，服务器群可以得到更多的部署。

4）尽管服务器群和分层缓存法通过使用缓存代理服务器可以解决网络性能问题，但它们也有不足之处。首先，由于服务器被部署在源服务器附近，当网络堵塞时它们对提高网络性能所起到的作用就会很小。缓存代理服务器或许在这种情况下有用，但它们是根据用户需求进行缓存，这可能会迫使拥有热点内容的内容提供商使用更大的服务器群、负载均衡和高速宽带连接去满足用户需求。为了解决这些不足，另一种内容网络于 20 世纪 90 年代开发出来，被称为内容分发网络（Content Distribution Network 或 Content Delivery Network），这种计算机网络系统利用 Internet 实现透明化协作，将内容传输到终端用户。

随着 CDN 的引入，内容提供商开始将他们的网站连接到 CDN。通过 CDN 进行内容服务后，他们就很快意识到，无须维护那些昂贵的基础设施就可以得到较高的可靠性和可扩展性。因此，提高 CDN 设施建设找到了充足的理由。Akamai Technologies[1, 27]诞生 MIT 对瞬时拥塞的研究成果，当时科学家们开发出了一系列突破性算法，实现在由遍布全球的服务器构成的大型网络上智能地进行内容的路由和复制。在近几年里，一些企业成为提供快速、可靠内容服务的专家，CDN 为行业带来了许多盈利的机会。瞬时拥塞问题[14, 34]（如美国 9·11 事件[10]）会造成某些网站的严重缓存问题。因为 CDN 可以有效地应对瞬时拥塞问题，许多 CDN 提供商在基础设施建设上加大了投资力度。第一代 CDN 主要将关注放在静态和动态 Web 文档[36, 61]；而第二代 CDN 开始着眼于用户互动性高的视频点播、新闻点播以及音频、视频等流媒体。同时，新一代 CDN 还可以把内容分发到移动设备上。然而，新一代 CDN 的研究大部分还处于探索阶段，并没有真正投入市场。我们预计，第三代 CDN 将以互联网社区为基础，也就是说，它将基于互联网用户群的共性。更多有关社区型 CDN 的信息可参见第 15 章。图 1.4 展示了 CDN 的演化过程以及对未来几年发展方向的预测。

随着 CDN 业务的发展，许多标准化组织开始出现，以满足行业的标准化需要。作为官方机构，互联网工程任务组（IETF）已采取多项措施，如对 RFC 文件的发布[15, 16, 24, 26]。除了 IETF，其他组织（如 BST[3]、ICAP 论坛[6]、互联网流媒体联盟[7]）也已采取相关措施来研究分发高宽带内容、音频视频等流媒体内容以及相关数据内容的互联网传输标准。从 2002 年开始，大型 ISP 开始自建 CDN 以提供定制化服务。

1.3.2 相关系统

数据网格（data grid）、分布式数据库和 P2P 网络这三种分布式系统与 CDN 服务有许多共同点。本节将从需求、功能和特性三方面对以上三种系统进行描述。表 1.1 对 CDN 和以上三种系统进行了比较，展现了其各自独特的性能。

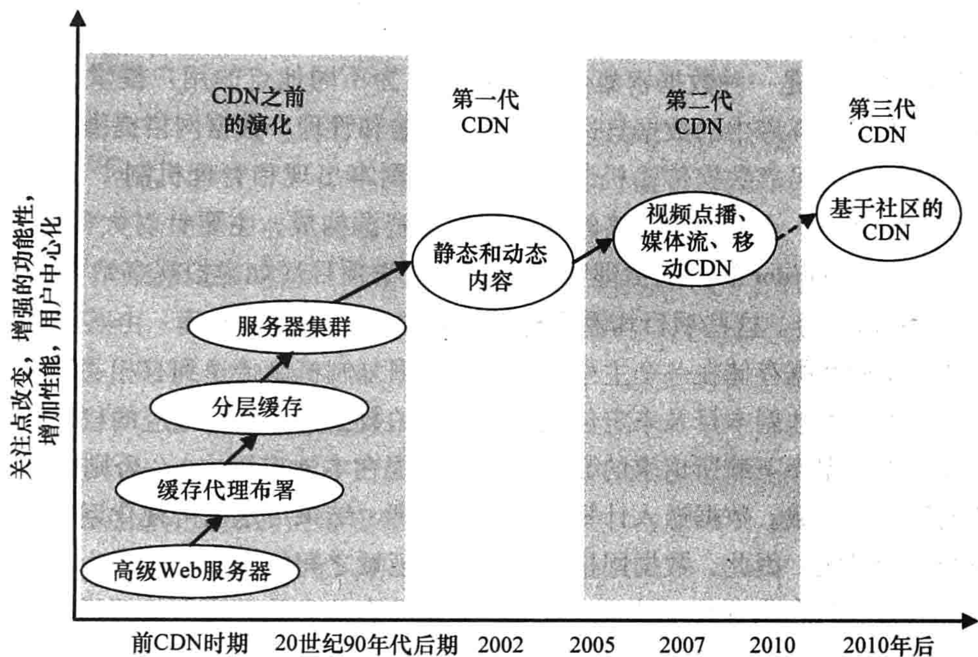


图 1.4　CDN 的演化

表 1.1　CDN 和相关系统之间的比较

特征	CDN	数据网格	分布式数据库	P2P 网络
分类	一组分布在互联网上的联网计算机集合	数据密集型计算环境	分布在多物理环境下的数据逻辑集合	通过资源对等聚集而形成的信息检索网络
组成	网络边缘上的缓存服务器	参与单位值对象的形成 注：[（VO）值对象]	现有数据库的联合与拆分	对等之间的协同
主要目的	在内容发布同时降低 Web 延时	通过预设置，最优资源选择，高速数据移动得到数据分布的性能增益	现有数据库和数据库碎片副本的透明集成	对等之间的文件共享
完整性	缓存之间的完整性	数据网格副本之间的完整性	多数据库之间的完整性	N/A
一致性	副本内容之间缓存的强一致性	数据网格副本之间的弱一致性	分布式数据库之间强一致性	缓存内容之间弱一致性
自治性		自治参与者	自治 DDB 网站	自治对等
业务活动	内容缓存	跨组织和跨界的数据无缝分析、协同与维护	查询检索处理、优化和管理	定位或缓存内容，加密、检索解密，并验证内容
管理	个体公司，所有权性质	机构联合体	单一授权实体	自利的端用户/对等

1. 数据网格

数据网格[43,62]是一种数据密集型计算环境，为不同地点的用户提供服务，可以对存储在分布式资料库中的数据集进行挖掘、传输和管理。数据网格提供两种最基本的服务：高性能的可靠数据传输机制与可扩展的副本出现和管理机制[22]。数据网格由分布在不同地点、通过高速连接的计算和存储资源构成，主要针对大型科研应用，如 Large Hadron Collidor 的高能物理实验[37]、天文学项目（如虚拟观察站[59]）和蛋白质模拟（BioGrid[2]）。这些项目均需要对海量数据进行分析和处理。由实验仪器和网络传感器产生的数据存储在一个主站点，通过数据复制机制传送到有相关需求的存储站点，用户通过本地副本目录来定位他们所需要的数据集。通过相应的权限许可，用户可通过本地数据库下载所请求的数据（如果数据在本地有副本），否则需要连接到远程数据库进行下载。数据送入计算单元进行处理，结果可送往可视化系统、共享数据库或个人计算机。因此，数据网格通过系统和区域之间的无缝衔接，为用户提供了一个分析数据、与合作者共享结果以及维护数据状态信息的环境。数据网格中的资源通常是多源的，并可能跨越多个管理域。因此，数据网格的特色在于海量数据的可视化、分布式数据集之间的共享、统一的命名空间逻辑结构以及数据管理。除此之外，数据网格也具有应用特性。数据网格的目标是将现有的分布式资源进行整合，通过数据共享和发布获得性能的提升。机构之间往往在某种共同利益的驱使下，形成一个虚拟组织来共享资源，这就构建了一个数据网格。与之不同的是，CDN 的主要目的在于对数据进行缓存，以加速端用户的访问。另外，所有的商业 CDN 都是私营性质的，它们由企业进行管理和运营。

2. 分布式数据库

分布式数据库[21,45]是分布在不同物理位置数据的一个逻辑组织集合，它可能是同一地点不同计算机中的数据，也可能是互联的计算机网络中分散的数据。分布式数据库中的每一台计算机都是一个节点，每个节点都可以充当客户端、服务器或身兼二职，这取决于所处的环境。每个站点都有一定的自主性，可以执行一个本地查询，也可参与执行一个全局的查询。分布式数据库可以由一个分拆的数据库组成，也可以由几个现有的数据库组合而成。这种系统的分布性对用户来说是透明的，因为它在与用户交互时表现为一个单一的逻辑系统。分布式系统内处理的事务（transaction）也是透明的，每个事务必须在多个数据库之间保持完整性。分布式数据库主要服务于大型企业，用来取代过去的集中数据库系统，实现现有数据库之间的互联，当新的组织单位加入时增加新的数据库。分布式数据库提供的应用包括分布式传输处理、分布式任务优化和资源有效管理。分布式数据库致力于整合已有的多种数据库，以获得一个统一、标准的查询接口，从而增强可靠性和吞吐量。分布式数据库的内部整合是由一个独立的组织来完成。与分布式数据库相类似，整个 CDN 也是由一个独立的自治实体来管理。然而，CDN 与分布式数据库的不同在于，CDN 的缓存服务器并不像分布式数据库那样有自治特性。此外，CDN 的目的是内容缓存，而分布式数据库主要用于

查询、优化和管理。

3. P2P 网络

P2P 网络[13,44]的目的是对计算机资源的直接共享，而不是借助于某个中转或管理中心。P2P 网络本质上是一个信息检索网络，通过 ad - hoc 将资源进行整合，形成一个部分的或完全的非中心化网络系统。在 P2P 网络系统内，每个成员均是自治的，并依赖其他对等成员获取资源、信息和转发请求。理想情况下，P2P 网络不存在控制中心，因此任何一个任务通常都由所有实体协同完成，如对其他节点的搜索、对内容的定位和缓存、对请求的路由、对内容的加密解密、检索和验证等。相比传统的中心化系统，P2P 网络系统具有更强的容错能力和可扩展性，因为它们不存在单点失效（Single Point of Failure）问题⊖。P2P 网络的一个成员可以在任意时间加入或离开该网络。P2P 网络更适合这样的内容提供商，如果它们难以接入或承受一般商业化 CDN 系统的价格。BitTorrent[33]是 P2P 系统的一个典型例子，它是一个非常流行的 P2P 文件分享应用。P2P 文件共享网络的目的在于制定有效策略使得能够在一个对等群体中定位特定的文件，并在网络不稳定的状态下为这些文件提供可靠的传输服务。此外，还可以管理由于大量访问流行内容带来的突发流量问题（如 flash crowds）。与 P2P 网络相反，CDN 的主要目的是关注用户的访问需求，而不侧重用户之间分享文件和内容的有效机制。同时，CDN 与 P2P 网络的另一个不同之处在于，CDN 对单位时间里加入和离开的节点数并不关心，而在 P2P 网络中这个数字非常重要。

1.4　对 CDN 的深入探究

从以上的讨论可以看出，CDN 的主要受益者是那些希望确保对终端用户服务质量的内容提供商。现阶段对 CDN 的分析表明，CDN 侧重于以下几种商业性参数指标：可扩展性、安全性、可靠性、快速响应以及性能。

1. 可扩展性

可扩展性指一个系统为了能够处理更多的数据、更多的用户和更好的传输，而对系统自身性能进行拓展的能力。为了进行全球尺度上的扩展，CDN 提供商需要投入时间和资金来构建额外的网络连接和设施，包括提供资源动态地应对瞬时拥塞和流量变化。当遇到 flash crowds 时，CDN 可充当流量的“减振器”，通过自动地按需提供网络容量，以此解决瞬时拥塞问题。只有具备这种能力，CDN 才能避免资源的过度配置，并为每个用户都提供高性能的服务。

在目前 CDN 的商业架构中，内容提供商付费给 CDN 提供商来最大化其提供内容

⊖ 单点失效（Single Point of Failure，SPoF）又称单点故障，是指一个系统中的这样一个部件：如果它失效或停止运转，将会导致整个系统不能工作。如电影《2012》中的那个齿轮，它被卡住后导致整个舱门无法关闭，进而使整个引擎无法启动。——译者注

带来的影响。然而，对现有 CDN 的趋势分析表明，应用类型的改变将会导致未来 CDN 模式的改变[53]。将来，内容提供商和终端用户还将会付费来获取高质量服务。而在这种状况下，可扩展性将是以低成本提供高质量内容服务的关键所在。

2. 安全性

CDN 的一个核心服务是为保密数据和高价值内容提供安全的分发方案[19]。这里所说的安全是相对于无授权访问和修改而言的内容保护。没有适当的安全控制，CDN 平台会受害于网络故障、分布式拒绝服务攻击（DDoS）、病毒以及其他意想不到的使 CDN 业务瘫痪的入侵。CDN 需要严格满足物理基础设施、网络、软件、数据和处理过程安全要求。一旦 CDN 的安全标准合格，则无须昂贵硬件和专门部件来保护内容的传输。CDN 提供商需要应对各种潜在的风险，包括 DDoS 和其他恶意活动。

3. 可靠性、快速响应和性能

可靠性是指对服务可用时段和服务不可用的条件具有可预期性，CDN 提供商通过从多个数据源传输特定内容来提高用户的访问质量。对于 CDN 这类需要极强容错能力的网络，适当的网络负载均衡机制是必需的[42]。快速响应通常指当遇到可能出现的服务中断情况时，服务器将需要多长时间才能恢复运转。CDN 性能通常以终端用户所获得的响应时间（如延迟）来衡量。响应时间过慢是用户关闭网页和程序的主要原因。CDN 的可靠性和性能是受到分布式内容地理位置和路由机制，以及数据复制和缓存策略的影响。因此 CDN 往往使用缓存和流式技术来提高媒体内容的分发性能[57]。CDN 还着重于提供快速、高可靠性的网页服务，以此来强化 CDN 的可靠性和以用户为中心。

1.5 CDN 的发展现状

本节将介绍目前 CDN 领域的发展水平，同时还将描述目前 CDN 的不同服务及相关技术。首先，简单介绍存在于内容分发领域的商业 CDN（如 Akamai、EdgeStream、Limelight Networks 和 Mirror Image）。随后，介绍学术 CDN（如 CoDeeN、Coral 和 Globule），这有助于读者了解目前 CDN 技术的全貌。

1.5.1 商业 CDN

绝大多数正在运营中的 CDN 是由商业企业开发的，这些公司大多经过多次兼并、收购和整合而成。因此，本节只着重研究那些在很长一段时间内有着稳定经营模式的公司。表 1.2 显示了四个商业 CDN，并附有简单介绍。目前大多数商业 CDN 的最新情况可以在 Davison[25] 和 Pathan [49] 的研究中找到。

表 1.2 现存商业 CDN 的汇总

名称	服务类型	覆盖情况	产品/解决方案（如果有）
Akamai www.akamai.com	提供 CDN 服务，包括流式技术	覆盖 85% 的市场，25 000个服务器、69 个国家、900 个网络处理占互联网总流量的 20%	Edge Platform（处理静态、动态内容），Edge Control（管理应用）网络操作控制中心（NOCC）

（续）

名称	服务类型	覆盖情况	产品/解决方案（如果有）
EdgeStream www. edgestream. com	在公共互联网上提供中断视频流应用	提供视频流（通过用户的电缆或 ADSL 调制解调器连接），即使在具有 20 个跳的路径上（从服务器到端用户）也能提供服务	EdgeStream 视频点播和 IPTV 流软件（供视频流使用）
Limelight Networks www. limelightnet works. com	提供分布式视频点播、直播音乐、游戏和下载	边缘服务器分布在全球 72 个地点	Limelight Content – Edge（通过 HTTP 进行分布式内容分发），Limelight MediaEdge Streaming（通过流式技术进行视频和音乐的分发），Limelight Custom CDN（用户分布式分发解决方案）
Mirror Image www. mirror – image. com	提供内容分发、流媒体、Web 计算和报表服务	边缘服务器分布在 22 个国家	全球内容缓存、Extensible Rules Engine（XRE）、VoD 及 Live Webcasting

商业 CDN 的私营性质使我们很难找到其技术和商业策略方面的详细信息。在这个商业 CDN 的发展状况调研里，我们提供了尽可能详细的信息。这里需要说明的是，许多 CDN 的特定信息，如 CDN 计费、已有用户等，都将被忽视不提，因为它们很可能是随时间快速变化的。因此，本节提供的信息主要是那些较为固定的和最新的发展状况。不过，如果读者想了解 CDN 定价机制，建议读者阅读本书第 8 章，那里概述了 CDN 服务的计费策略。

1. Akamai

Akamai[1, 27]于 1998 年在美国马赛诸塞州成立。它由 MIT 的一个针对瞬时拥塞的研究发展而成。Akamai 是内容分发服务领域的领跑者，在 69 个国家的 900 个网络内拥有超过 25000 台服务器[1]。Akamai 的技术基于以下事实，即从单一地点提供网页内容服务会导致网站可扩展性、可靠性以及性能的严重下降。因此，Akamai 可以被视为一个从位于网络边缘的多个缓存服务器进行服务的系统。Akamai 服务器可以分发静态数据（如 HTML 网页、图像、可执行程序和 PDF 文档）、动态内容（如动画、脚本和 DHTML）以及流式音视频。

Akamai 的架构在处理瞬时拥塞问题时，把更多的服务器分配给高负载网站，让较近的服务器来服务用户。系统将用户请求分配到最近的可用服务器以获取所需内容。Akamai 通过映射（mapping）技术（也就是将请求定向到内容服务器），即使用动态的具有容错能力的 DNS 提供自动的网络控制能力，映射系统根据被请求的服务、用户位置和网络状态来解析主机名，并使用 DNS 实现网络负载均衡。Akamai 的域名服务器通过将请求映射到相应的服务器来将主机名解析为 IP 地址。Akamai 代理节点与特定的边界路由器进行对等通信；映射系统使用 BGP（Border Gateway Protocol）[56]

信息确定网络的拓扑结构。Akamai 的映射系统结合网络的地域信息以及网络实时统计数据（如 traceroute 数据[39]）来提供详细、动态的网络结构视图和不同映射系统的服务质量测量。

Akamai 基于 DNS 的负载均衡系统持续监控服务状态以及服务器和网络状况。为了监测整个系统的运转状况，Akamai 使用代理，通过下载网络对象并测量该过程中的出错率以及下载次数来模拟终端用户的行为。Akamai 利用这些信息监测整个系统的性能，自动检测并暂停存在问题的数据中心和服务器。每个内容服务器不断向一个监控应用报告其负载状况，该应用将这些数据进行汇总并向本地 DNS 服务器报告。然后，DNS 服务器由此来确定当解析主机名时应该返回哪些 IP 地址（2 到多个）。如果某个服务器的下载量超出了一个界限，那么 DNS 服务器会及时将该服务器的部分工作分配给其他服务器。如果这个服务器的下载量再次超出界限，则这个服务器的 IP 地址对于用户将不再可用。这样，当服务器的负载从适度增加到较高时，服务器可以分流部分负载。Akamai 监控系统也会将数据中心的负载转移到顶层的 DNS，从而将流量从过载的数据中心处引开。除了负载均衡，Akamai 的监控系统为每个用户和内容服务器提供内容服务的集中报告，这些信息对于网络运营和诊断非常有用。

Akamai 通过 HTTP 和 HTTPS 传输静态和动态内容。Akamai 内容服务器利用生存期等特性（如使用 HTTPS 协议提供安全内容服务的能力、支持内容替换、编码转换、cookie 使用等）来支持相关的静态内容的分发。基于这些特性，边缘服务器保证了内容的一致性。另外，Akamai 利用 Edge Side Includes（ESI）[4] 在边缘服务器上处理动态内容。利用 ESI 可以使内容提供商将其动态内容分成片段，这些内容片段都具有独立的缓存能力，在 Akamai 的边缘服务器上可被作为分散的对象来维护，并根据终端用户的请求，动态地组装呈现在动态网页上。

Akamai 支持微软 Window Media、Real 和苹果 QuickTime 格式的流媒体传输服务（实时媒体和媒体点播）。实时流媒体由内容提供商产生和编码，并发送到一系列 Akamai 边缘服务器的接入点，从而将内容提供给最终用户。为了避免所有的单点失效，数据的备份将被保存在接入点服务器。此外，接入点服务器使用信息分散技术（information dispersal techniques）通过多个冗余路径将数据发送到边缘服务器。

有关 Akamai 和覆盖路由的更多信息（包括其性能、可用性、实时流的不同应用和 IP 加速等）可参阅本书第 10 章内容。

2. EdgeStream

EdgeStream[23] 于 2000 年在美国加利福尼亚成立。EdgeStream 是一个提供互联网视频流应用的服务公司，主要提供视频点播和 IPTV 流软件，可以实现互联网高码率视频的传送。它使用 HTTP 流进行内容分发。EdgeStream 公司支持多种不同的视频压缩格式以供内容分发使用。该公司已开发出一系列技术，如连续路由优化软件（CROS）、网络拥塞隧道穿越（ICTT）以及实时性能监控服务（RPMS）等。这些技术有助于解决延时、丢包和网络阻塞等问题。在消费电子设备、无线手持终端、IP 电视机顶盒以及先进数字电视的嵌入式应用中，都能够使用 EdgeStream 进行视频流服务。

EdgeStream 的平台包括客户端和服务器软件两部分。服务器软件包括内容管理和

在线报告（CMOR）服务器软件模块、EdgeStream 控制服务器软件模块、EdgeStream 数据库系统模块以及 EdgeStream 流服务器模块。所有服务器模块既可以组合在一台服务器上运行，也可以单独运行。CMOR 模块负责管理账号、内容和系统中所有其他服务器。CMOR 也可通过 SQL 数据库生成基于 Web 的统计数据和事务处理的实时报告。控制模块提供必要的授权以获得内容的位置信息和流式数据分布管理以及日志功能。数据库模块负责维护涉及记账和收费的日志。EdgeStream 使用微软的 SQL 2000 标准版或企业版服务器软件。流服务器模块进行了负载均衡优化设计，可在低成本的标准服务器平台上运行。当在双处理器的服务器上运行时，流的传输速度可以超过 500Mbit/s，同时具备 TB 级的存储能力。

EdgeStream 客户端软件提供了对于 Windows 媒体播放器和 Real Player 的插件接口。同时，该软件也能用于检测网络连接的质量。PC 客户端软件是基于标准的 Windows 平台，仅有 600KB 大小。Firmware 客户端软件仅有 300KB 左右（或更小），并可以嵌入到 Windows XP 或者 Windows CE 上。

3. Limelight 网络

Limelight 网络[30] 于 2001 年创建于美国亚利桑那的坦佩。Limelight 网络提供的内容分发服务包括数字媒体内容的 HTTP/Web 分发，如视频、音频、游戏、软件和社交媒体等。它将内容发布给媒体公司，包括电视传媒、音乐、广播、新闻、杂志、影视、视频游戏和软件企业。

内容提供商将内容直接上传到 Limelight 的网络服务器，或上传至自己的服务器上，这些服务器和 Limelight 网络互联。当收到来自端用户的请求时，Limelight 将内容发送到一台或多台 Web 服务器集群，这些集群会将内容转发至全球每一个经过预先配置的服务器处。然后，内容将直接通过 ISP 或者通过合适的公网发送到端用户。同其他商业 CDN 相同，Limelight 也是用 DNS 重定向技术来将用户请求重新路由至本地服务器集群处。通过结合 BGP 反馈和自身的测量（如从大量可用位置得到的 trace route 信息），Limelight 创建了详细的互联网映射关系。

Limelight 网络提供流媒体点播分发服务时，支持 Adobe Flash、MP3 音频、微软的 Windows Media、Real 以及苹果公司的 QuickTime 媒体格式。Limelight 网络的专有软件 Limelight ContentEdge 使用 HTTP 进行分布式内容分发，Limelight MediaEdge 使用流媒体技术进行分布式视频和音频内容的发布，Limelight StorageEdge 供用户在 Limelight 的 CDN 架构内存储自己的内容，同时 Limelight 的定制 CDN（Custom CDN）可提供定制化的分布式分发解决方案。需要使用 Limelight 流技术服务的内容提供商会使用 Limelight User Exchange（LUX），这是一个基于 Web 的管理和汇报控制台，用于跟踪并实时报告端用户的行为。所有软件汇聚在一起，将有助于管理内容分发系统。

4. Mirror Image

Mirror Image[40] 于 1999 年成立于美国马萨诸塞州，是一个向用户提供在线内容、应用程序、流媒体、Web 计算、新闻报告、交易传递服务的提供商。Mirror Image 采用集中式的“超市”架构，其中内容被存放在一个极大的 Web 服务器集群中，集群位于稠密用户区的中心地带。Mirror Image 利用运行在互联网之上的一种全局内容接

入点设施架构（CAP）来提供内容发布平台，主要面向内容提供商、服务提供商和各种企业。

当一个用户通过 Mirror Image 配置的站点请求服务时，该请求会被自动路由到 CAP 网络上的一个全局负载均衡器上。该负载均衡器使用 DNS 路由来确定具有最快响应时间的 CAP 位置。在选定的 CAP 位置处，系统从缓存和核心数据库中搜索被请求的内容。如果内容找到，便发往用户；否则，CAP 网络会自动将一个重定向状态代码“302”返回到源服务器，由源服务器将内容发往用户处，CAP 网络同时也从源服务器检索或“拉取”内容，并将其存储以供下次使用。

Mirror Image 提供内容分发、流媒体和 Web 计算方案，包括全局内容缓存方案，用于服务静态内容时的流量分流。数字产品下载方案用于管理存储和下载内容，视频点播业务用于内容的流发布。可拓展规则引擎（XRE）给予用户关于分发过程的控制权，网播方案允许用户发送一对多的消息，这给培训、市场和远程教育提供了潜在机遇。

1.5.2 学术 CDN

不同于商业 CDN，P2P 技术的使用在学术 CDN 中较为普遍。因此，内容分发遵循去中心化的方法且请求负载遍布在所有参与主机上，这就使系统能够有效应对节点失效和突发性的负载激增。需要注意的是，基于 P2P 技术的学术 CDN 仅对静态内容有效，不能处理动态内容，因为动态内容本质上是不能缓存的。

在本节中，给出了三种具有代表性的学术 CDN，即 CoDeeN、Coral 和 Globule。表 1.3 提供了这些学术 CDN 的简要资料。其他两个学术 CDN，即 FCAN（用于缓解瞬时拥塞的自适应 CDN）和 COMODIN（用于协作媒体流服务的流 CDN），将在本书的第 11 章和第 12 章分别给出介绍。

表 1.3 现有学术 CDN 汇总表

名称	描述	服务类型	实现与测试	可获得性
CoDeeN www.codeen.cs.princeton.edu	一个学术 CDN 测试平台搭建在 PlanetLab 上的	提供内容缓存和 HTTP 请求重定向	使用 C/C++实现，在 Linux（2.4/2.6 内核）和 MacOS（10.2/10.3）上测试	N/A
Coral www.coralcdn.org	一个免费 P2P CDN，由 PlanetLab 托管	提供和流行度相关的内容复制	使用 C++实现，在 Linux、OpenBSD、FreeBSD 和 Mac OS X 上测试	没有官方发布版 Coral 是免费软件，遵循 GPLv2（GNU 通用公开许可）
Globule www.globule.org	一个开源的协作式 CDN	提供内容的复制和服务器监控，并重定向用户请求到可用副本服务器	使用 PHP 脚本 C/C++实现，在 UNIX/Linux 和 Windows 上测试	Globule 是开放源码软件，遵循 BSD 类授权及 Apache 软件授权（对于打包的 Apache HTTP 服务器）

1. CoDeeN

CoDeeN[46, 64]是一个基于 P2P 的代理服务器系统，由普林斯顿大学开发。它基于 HTTP，目的是使相关参与用户在访问大多数 Web 站点时获得更好的性能。CoDeeN 提供了 Web 内容的缓存和 HTTP 请求的重定向。CoDeeN 建立和运行在 PlanetLab 之上[9]，是一个由一系列高性能代理服务器构成的网络。CoDeeN 代理服务器的部署为“开放式”，能够允许外部的托管组织访问。每一个 CoDeeN 节点都能作为正向代理、反向代理或者重定向器使用。CoDeeN 通过如下方式运行：①用户在 CoDeeN 的一个宽带代理服务器上设置他们自己的网络缓存，该代理服务器通常要求距离用户很近；②CoDeeN 的节点作为正向代理服务器使用，试图满足本地请求。如果出现缓存未命中的情况，CoDeeN 节点上设置的重定向器逻辑单元会判断请求应该被送往何处。对于大多数请求而言，重定向器将结合请求的本地性、系统负载、可靠性和邻近性来将请求转发到其他的 CoDeeN 节点，这些节点将作为反向代理服务器处理请求。如果请求仍无法得到处理，则将其转向源服务器。

CoDeeN 具有本地监控能力，能够监控服务器的基本资源情况，如可用文件描述符/套接字、CPU 负载和 DNS 解析服务等。它能够收集 CoDeeN 的系统状态和主机环境，这些信息有助于对资源连接和外部服务的可用性做出评估。为了监控对等成员的可用性和状态，每一个 CoDeeN 节点都使用两种机制：一个轻量级的 UDP 心跳信息和一个重量级的 HTTP/TCP 层的接收器[64]。第一种情况中，每一个代理服务器每秒发送一个心跳报文到网络中的一个对等成员，该成员回复（心跳确认或者 ACK）所承担的负载信息，包括对等成员的平均负载、系统时间、文件描述符可用性、代理服务器和节点的正常运行时间、每小时的平均流量及 DNS 计时/故障统计数据等。通过将 ACK 的历史记录与它们所承载的本地状态信息相关联，每一个 CoDeeN 节点会对其他节点的健康状况进行独立评估。在第二种方法中，每一个 CoDeeN 节点都被赋予一个成本值，用于衡量该节点在规定时间内无法获取一个页面时的失败情况。每一个对等成员未能完成的页面读取历史记录都被维护起来，并与 UDP 层的心跳报文相结合，能够有助于判定某个节点是否能用于请求重定向服务。

很多项目都与 CoDeeN 紧密相关，如 CoBlitz（一个针对大文件的基于 Web 的分布式系统）、CoDeploy（一个高效的同步化工具，用于 PlanetLab 的“片段”之间同步）、CoDNS（一个快速、可靠的名字查找服务）、CoTop（一个 PlanetLab 网络中基于命令行的行为监控工具）、CoMon（一个基于 Web 的“片段”监控器，用于监控大多数 PlanetLab 网络的节点）和 CoTest（一个登录检测工具）。

CoBlitz 是一个运行在 CoDeeN 之上的重要应用服务[48]，其实质是一个文件传输服务，可用于在不需要对标准 Web 服务器和用户进行任何修改的前提下分发大文件（因为所有分发需要的支持都能在 CoDeeN 上得到）。在 CoDeeN 上建立 CoBlitz 的动机之一是为了确保数据的长期缓存，使得即使当请求失效之后用户请求的内容仍可以被迅速服务。CoBlitz 可以公开访问，允许用户使用源地址“http：//coblitz. codeen. org：3125”登录访问。一个定制化的 DNS 域名解析服务器将域名 coblitz. codeen. org 解析

为用户附近的某个 PlanetLab 节点。为了部署 Coblitz 这个基于 HTTP 的 CDN，需要修改 CoDeeN，使之能够处理大数据文件。这个方法中包括了对大数据文件的修改，以便使其能有效地应用在 CoDeeN 上，同时也包括修改它的请求路由算法，这样就能处理过载情况下的海量访问。在 CoBlitz 中，一个大数据文件被认为是由一组小文件（或文件片段）组成，能够在 CDN 上存储并发布。这些片段是否全部缓存，部分缓存，或根本没有缓存，CoBliz 依然可以正常工作，如需要可以从源服务器拿到丢失的片段。因此，在使用 CoBlitz 传输大文件时，不必考虑任何对等成员上是否存有该文件。

2. Coral

Coral[28]是一个免费的 P2P 内容分发网络，由纽约大学的“安全计算机系统”研究小组在访问斯坦福大学时开发。它的设计思路是对 Web 内容作镜像服务，为大多数用户带来良好的 Web 站点访问性能。Coral 利用自愿加入者的带宽来避免瞬时拥塞，并降低 Web 站点和其他 Web 内容提供商的负载。Coral CDN 部署在 PlanetLab 之上，而非第三方的系统。为了使用 CoralCDN，一个内容发布者、终端用户或者其他任何使用者需要在 URL 的主机名后面加上“.nyud.net:8090”的后缀。DNS 重定向方法以透明的方式将用户重定向到附近的 Coral Web 缓存。Coral 的 Web 缓存彼此合作，从附近的对等成员处传输数据，借此最小化源服务器的负载和降低用户感知到的延时。CoralCDN 建立在 Coral 的关键值检索层之上，使节点可以在附近的节点上访问被缓存的对象，无须额外地检索更远处的其他节点。同时，它避免了在检索设施中出现“热点”，甚至在负载减弱的情况下也能实现这一功能。

CoralCDN 主要由三部分组成：①协作式 HTTP 代理服务器组成的网络，用于处理用户请求；②一组 DNS 域名解析（即“.nyud.net”）服务器系统，用于将用户映射到邻近的 CoralCDN HTTP 代理服务器；③一组底层的检索设施和集群机制，这是前两个部分的基础。

Coral 使用一个检索摘要，称为分布式散列哈希表（DSHT），这是一个分布式哈希表（DHT）的变体。DSHT 设计用于存储软状态“Key - Value”对的应用，其中一个关键字可以对应多个值。DSHT 将“Key - Value”对缓存在那些标识符接近关键字的节点处，作为负载的反函数。它使用“散列的”存储技术，有利于路由层和存储层之间的跨层交互。

Coral 对 URL 完成修改后，针对这些修改后的 URL 的 HTTP 请求由 CoralHTTP 代理服务器来处理。为了最小化源服务器的负载，一个 CoralHTTP 代理服务器会尽可能地从其他代理服务器处读取页面。每一个代理服务器都保留一份本地缓存来响应实时请求。如果一个 CoralHTTP 代理服务器发现被请求内容存在于一个或多个其他代理服务器，则它与这些服务器之间同时建立 TCP 连接，并向它能连接上的第一台代理服务器发出一个 HTTP 请求。一旦邻近的代理服务器开始发送有效内容，其他所有已经建立的 TCP 连接都将被关闭。当一个用户从非本地 URL 处请求内容时，CoralHTTP 代理服务器会首先尝试本地进行副本缓存。如果 Coral 的检索层无法提供任何参考信息或者没有一台代理服务器能够返回被请求的内容，那么 CoralHTTP 代理服务器就从源

服务器上直接读取内容。在面对瞬时拥塞时，所有的 CoralHTTP 代理服务器都自发地组成一种“多播树”，用于检索 Web 内容。然而，代理服务器并不同时对源服务器发送检索请求，数据从在源服务器上读取内容的代理服务器处向那些后到的代理服务器发送请求数据。在 CoralCDN 上，该方式考虑了联合乐观参考路由和切入路由选择的方法。

CoralDNS 服务器将经过 Coral 化的 URL 主机名解析成 IP 地址，并返回给 CoralHT-TP 代理服务器。Coral 的架构是基于连接良好的计算机集群。集群面向高层的软件的接口，构成了 DNS 重定向机制中的一个重要部分。为了提升系统的本地性，当一个 DNS 解析器试图与邻近的 CoralDNS 服务器通信时，CoralDNS 服务器将一个适当集群内的代理服务器返回给它，以确保今后该用户的 DNS 请求都不会离开这个集群。一个 CoralDNS 服务器仅仅返回 CoralHTTP 代理服务器的地址，而这些代理服务器是最近被验证通过的，目的是在响应 DNS 检索请求之前同步检测代理服务器的健康状态。

3. Globule

Globule[52]是一个开源的协作式 CDN，由阿姆斯特丹的 Vrije 大学开发。Globule 的目的在于允许 Web 内容提供商能够自我组织并运营它们自己的广域托管平台。具体而言，Globule 是一个覆盖网，由通过广域网以 P2P 方式互联的端用户节点组成，其成员提供各种资源，如存储空间、带宽、处理器等。它提供了对内容的复制、对服务器的监控以及将用户请求重定向到可用的副本服务器。

在 Globule 中，一个站点被定义为一组文档的集合，属于某个特定用户（站点的所有人），而一个服务器就是连接到网络上的计算机中运行的进程，执行着 Globule 软件的一个实例。每一个站点都由源服务器、备份服务器、副本服务器和重定向服务器组成。源服务器拥有该站点的所有文档，并负责向其他相连的服务器分发内容。备份服务器维护着一个托管站点内容的即时更新的完全副本。不同于备份服务器，很多副本服务器可以被用于托管站点。备份服务器仅仅维护内容副本，而副本服务器的目的是基于请求负载和服务质量（QoS）的需要来最大化性能。一个副本服务器一般由不同于源服务器的另一个用户负责运行，并且一个副本服务器一般负责维护托管站点的所有内容的部分副本。可以把副本服务器视为一个带缓存功能的代理服务器，负责在本地缓存未命中的情况下从源服务器取内容。一个重定向服务器负责将用户请求重定向到能够服务该请求的最优副本服务器处。Globule 的重定向器可以使用 HTTP 和 DNS 两种方式。一个重定向器也能实现一个用于用户重定向的策略。默认策略是根据预计的延时时间，将用户重定向至最近的副本服务器。重定向器也监控其他服务器的可用性，以确保有效的请求重定向。根据服务角色不同一台服务器可以是源服务器、备份服务器、副本服务器和重定向服务器中的任意一种。

Globule 将节点间延时作为副本服务器与用户之间邻近度的度量。这种度量用于将副本服务器最优地部署在用户附近，并将用户请求重定向到合适的副本服务器处。Globule 是作为 Apache HTTP 服务器的第三方功能模块实现的，能够允许任何服务器将其文档复制到其他 Globule 服务器。为了复制内容，内容提供商仅仅需要在它们的 Apache 服务器上再安装一个附加模块，并编辑一个简单的配置文件即可。

1.6 写给使用者

本节主要对未来CDN业界相关研究中可能实现的下列技术革命进行展望。

1. 一个统一的内容网络

为了使内容转换、处理以及基础设施服务方便易用，商家已经开发了内容服务网络（CSN）[38]，能够作为CDN之上的另一个网络体系架构层使用，并能够提供下一代的CDN服务。CSN可以被看成传统CDN的变体。CSN提供的网络资源被作为“服务”分发渠道供增值服务提供商使用，目的是把应用作为基础设施服务。在“内容分发”和“内容服务”的语境下，目前内容和服务从逻辑上是分离的，但如果考虑到内容网络的未来发展趋势，这种状况是不可取的。因此，一个统一的内容网络架构应当支持组件间的相互协作和内容以及服务的分发，这是未来的发展趋势。

2. 动态内容

一般而言，动态内容是指在使用Web应用时基于用户请求所产生内容的。这种内容的生成是高度定制化的，随用户背景和特性的不同而不同。大量的Web内容都是动态生成的，包括脚本、动画、DHTML或XML页面，随着用户的特定需求而随时产生。Web页面的动态生成可以通过使用可扩展的Web应用托管技术来进行（如前端计算[55]、内容感知的数据缓存[20,58]、数据复制[58]以及内容未知的数据缓存[58]）。不同于由Web服务器生成的动态页面的复制，这些技术的目的是在边缘服务器上复制和生成页面[58]。商业CDN提供商具有适合自身的解决方案和应用服务器平台来处理动态内容。Akamai公司的EdgeSuite和IBM公司的Websphere都是这些系统的典型例子，它们能够提供基于使用的应用和（动态）内容分发。为了管理动态内容，一个CDN提供商可能会使用这些可扩展的技术来加速Web页面的动态内容生成。所以相关策略的选择可能随Web应用的特性不同而不同。

3. Web服务

如今，已经有一部分商业CDN承载了Web服务。例如，Akamai已经在自己的网络上部署了.NET服务。Mirror Image也部署了应用分发网络（ADN），能够在自己的边缘服务器上同时托管.NET和J2EE的应用。若干研究成果[29,60]也已经表明，Web服务性能低下的主要原因就是存在对处理的需求和特定托管容量的问题。文献中已经提出了一些解决方案用于解决Web服务在边缘服务器上的高效复制问题。Geng等人[29]提出在网络和多ISP之间的可共享的和可交换的缓存架构。这个解决方案的特点在于设计一个容量配置网络（CPN），用于调整缓存容量。CPN由一个交互中心控制，而非某个特定CDN。该解决方案可以使CDN获取（通过交易）必要的容量来满足Web服务缓存的要求。Takase等人[60]提出了使用XML报文进行缓存，并通过缓存XML解析器的事件序列来得到性能提升。他们也提出了使用Java并发、反射复制和克隆复制的方法来缓存应用对象。

4. 面向服务的架构

内容网络的未来趋势是通过标准的协议和调用机制，允许一个服务的功能由其他

服务组合实现。因此，内容网络应该能够采纳面向服务的体系架构（SOA），这就需要 SOA 内部具有高度的透明性，从而对组件的融合技术产生影响。在这种基于 SOA 的 CDN 上进行内容管理，希望与用户偏好紧密相关。也就是说，在未来 CDN 中，全面综合的管理分布式内容和服务模型对终端用户选择所需的喜爱内容非常有帮助至关重要。为了解决这个问题，内容可以进行个性化定制，以满足特定用户（或者一组用户）的需求。如同 Web 个性化[41]一样，用户偏好可以通过数据挖掘技术自动地从内容请求和相关数据中获得。在内容网络上使用数据挖掘技术，基于 SOA 的 CDN 可以对流量、定价和记账/收费进行有效的管理，从而获得极大的性能提升。

1.7 未来研究方向

本节将分析 CDN 领域面临的机会和挑战，这些都有可能推动 CDN 技术的革新，在此基础上为 CDN 的研究者提供一个未来的发展方向。

1. 通过协作实现负载均衡和内容复制

无论传统的还是协作式 CDN 结构，有效的内容复制和缓存都有着至关重要的作用。缓存“热点”内容的概念并不算新，但在内容协作分发背景下，缓存“热点”将是一个非常重要且具有竞争力的方法。使得请求本地区响应，而且要关注内容最多请求的区域。未来的研究将逐渐趋向于动态、可扩展以及有效复制机制，该机制将面向用户需求。此外，我们希望出现将复制和缓存技术相集成的解决方案，这在协作式的方法将有效解决动态内容和个性化内容的管理问题。第 3 章针对这种内容复制技术提供了更多的信息。有关缓存复制技术的一体化应用以及缓存一致性机制的详细介绍将在本书的第 4 和第 5 章中给出。

2. 采用市场机制

经济模型可以用来开拓 CDN 市场的活力，通过分析新兴市场的行为使 CDN 具有更好的可管理性。同时，也为 CDN 提供资源和开放更多有兴趣的新服务带来更多的收益市场机制的采用可以通过 SOA 来实现。此外，复制、资源共享、负载均衡策略需要由盈利性作为引导，以满足终端用户的 QoS 要求。有关 CDN 的经济性设计和定价的信息见第 7 和第 8 章。

3. 一个针对流媒体的自适应 CDN

承载流媒体点播服务是非常困难的，因为将大量内容同时传输到终端用户意味着大量网络和带宽的需求。为了提高性能和避免网络拥塞，P2P 技术可以用来建设自适应的 CDN 结构。在该系统中，从流服务器、网络和存储资源分发到各终端用户内容和造成的工作负载被有效地分散开来，其根本思路是允许多个订阅者同时为访问相同视频的用户提供服务，而不是传统的单一服务器到客户端的流模式，此时，每个客户端只需存储一小部分内容。使用 P2P 方法的流媒体能够节省资金，已经用于非中心化流媒体架构（DeMSI）的设计[65]。另外，在开放和自适应流 CDN 中对媒体流服务进行协作控制的相关工作将在第 12 章进行介绍。

4. 移动动态 CDN

大量高流动性用户的信息分发使移动网络越来越受到关注。与有线网络相比，移动网络可以根据用户的流动性提供更强的潜在多样化服务。移动网络的内容需求在空间和时间上表现出很强的变化性，相应的分发技术必须考虑到这些特点，以实现系统的动态配置，并减小网络主干部分的总流量。移动动态 CDN 模型的设计应该力求解决用户的准确接入，信息及时更新及相关企业应用问题。这种用于企业网的移动动态 CDN 模型和相关的内容管理策略由 Aioffi 等人[12]提出。Ortiva Wireless[8]就是 CDN 提供商商业模型的一个案例，它可以向移动用户分发视频、音频以及多媒体内容。更多有关移动 CDN 信息传播的内容将在第 14 章介绍。

5. 通过互联/对等/代理的内容分发

目前内容网络的趋势和内容联网能力激发了人们对于内容网络互联的兴趣。CDN 之间通过相互协作可以获取高质量服务，这种协作提供了一种在不同 CDN 系统间重新进行内容分发的手段。引入协作后能够服务的用户数量远远超过单个 CDN 所能达到的数量。因此，未来研究将重点关注 CDN 之间的互联、代理以及对等组织等方面的技术创新[18,26,50]。CDN 互联的详细内容在第 16 章给出。

1.8 结论

本章向读者呈现了 CDN 的现状，并对目前应用于内容分发领域的技术进行了深入的分析。通过分析目前内容网络的趋势，可以预言，将新技术与现有技术（如代理、P2P、网格和数据挖掘等）相结合，可以提升未来 CDN 架构的性能。同时，我们发现 CDN 行业可能会发生一些变化，如由 CDN 互联、自适应 CDN、移动 CDN 向完全的、基于社区的 CDN 转变。因此，以本章的内容为铺垫，后面的章节中将向读者提供更深入的分析，使读者对内容分发领域的现状以及未来的趋势获得一个完整的理解。

参考文献

[1] Akamai Technologies, 2007. www.akamai.com
[2] BioGrid Project, Japan, 2005. http://www.biogrid.jp
[3] Broadband Service Forum, 2007. http://broadbandservicesforum.org
[4] ESI Developer Resources, 2007. http://www.akamai.com/html/support/esi.html
[5] IBM WebSphere Application Server, 2007. http://www-306.ibm.com/software/webservers/appserv/was/
[6] ICAP Forum, 2007. http://www.i-cap.org/
[7] Internet Streaming Media Alliance, 2007. http://www.isma.tv/
[8] Ortiva Wireless, 2007. http://www.ortivawireless.com/
[9] PlanetLab Consortium, 2007. http://www.planet-lab.org/
[10] Wikipedia. September 11, 2001 attacks. http://en.wikipedia.org/wiki/September_11,_2001_attack
[11] Adler, S. The slashdot effect: an analysis of three Internet publications. *Linux Gazette Issue*, 38, 1999.
[12] Aioffi, W. M., Mateus, G. R., de Almeida, J. M., and Loureiro, A. A. F. Dynamic content distribution for mobile enterprise networks. *IEEE Journal on Selected Areas in Communications*, 23(10), pp. 2022–2031, 2005.

[13] Androutsellis-Theotokis, S. and Spinellis, D. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4), ACM Press, NY, USA, pp. 335–371, 2004.
[14] Arlitt, M. and Jin, T. A workload characterization study of 1998 world cup Web site. *IEEE Network*, pp. 30–37, 2000.
[15] Barbir, A., Batuner, O., Beck, A., Chan, T., and Orman, H. Policy, authorization, and enforcement requirements of the open pluggable edge services (OPES). Internet Engineering Task Force RFC 3838, 2004. www.ietf.org/rfc/rfc3838.txt
[16] Barbir, A., Penno, R., Chen, R., Hofmann, H., and Orman, H. An architecture for open pluggable edge services (OPES). Internet Engineering Task Force RFC 3835, 2004. www.ietf.org/rfc/rfc3835.txt
[17] Bartolini, N., Casalicchio, E., and Tucci, S. A walk through content delivery networks. In *Proc. of the 11th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, LNCS Vol. 2965/2004, pp. 1–25, April 2004.
[18] Biliris, A., Cranor, C., Douglis, F., Rabinovich, M., Sibal, S., Spatscheck, O., and Sturm, W. CDN brokering. *Computer Communications*, 25(4), pp. 393–402, 2002.
[19] Brussee, R., Eertink, H., Huijsen, W., Hulsebosch, B., Rougoor, M., Teeuw, W., Wibbels, M., and Zandbelt, H. Content distribution network state of the art," Telematica Instituut, 2001.
[20] Buchholz, T., Hochstatter, I., and Linnhoff-Popien, C. A profit maximizing distribution strategy for context-aware services. In *Proc. of 2nd International Workshop on Mobile Commerce and Services (WMCS'05)*, pp. 144–153, 2005.
[21] Ceri, S. and Pelagatti, G. *Distributed Databases: Principles and Systems*, McGraw-Hill, NY, 1984.
[22] Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., and Tuecke, S. The data grid: towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23, pp. 187–200, 2001.
[23] Chung, R. Network latency and its effect on video streaming. EdgeStream, 2004. www.edgestream.com
[24] Cooper, I., Melve, I., and Tomlinson, G. Internet Web replication and caching taxonomy. Internet Engineering Task Force RFC 3040, 2001.
[25] Davison, B. D. Web caching and content delivery resources. http://www.web-caching.com, 2007.
[26] Day, M., Cain, B., Tomlinson, G., and Rzewski, P. A model for content internetworking (CDI). Internet Engineering Task Force RFC 3466, 2003.
[27] Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. Globally distributed content delivery. *IEEE Internet Computing*, pp. 50–58, 2002.
[28] Freedman, M. J., Freudenthal, E., and Mazières, D. Democratizing content publication with Coral. In *Proc. of 1st USENIX/ACM Symposium on Networked Systems Design and Implementation*, San Francisco, CA, USA, 2004.
[29] Geng, X., Gopal, R. D., Ramesh, R., and Whinston, A. B. Scaling web services with capacity provision networks. *IEEE Computer*, 36(11), pp. 64–72, 2003.
[30] Gordon, M. The Internet streaming media boom: a powerful trend that represents fundamental change. Limelight Networks, 2007. www.limelightnetworks.com
[31] Hofmann, M. and Beaumont, L. R. *Content Networking: Architecture, Protocols, and Practice*. Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 129–134, 2005.
[32] International Standards Organization (ISO), Open systems interconnection–basic reference model. ISO 7498, 1989.
[33] Izal, M., Urvoy-Keller, G., Biersack, E. W., Felber, P., Hamra, A. A., and Garces-Erice, L. Dissecting bittorrent: five months in a torrent's lifetime. In *Proc. of 5th Annual Passive and Active Measurement Workshop (PAM'2004)*, Antibes Juan-Les-Pins, France, 2004.
[34] Jung, J., Krishnamurthy, B., and Rabinovich, M. Flash crowds and denial of service attacks: characterization and implications for CDNs and Web sites. In *Proc. of the International World Wide Web Conference*, pp. 252–262, 2002.
[35] Kangasharju, J., Roberts, J., and Ross, K. W. Object replication strategies in content distribution networks. *Computer Communications*, 25(4), pp. 367–383, 2002.
[36] Lazar, I. and Terrill, W. Exploring content delivery networking. *IT Professional*, 3(4),

pp. 47–49, 2001.

[37] Lebrun, P. The large hadron collider, a megascience project. In *Proc. of the 38th INFN Eloisatron Project Workshop on Superconducting Materials for High Energy Colliders*, Erice, Italy, 1999.

[38] Ma, W. Y., Shen, B., and Brassil, J. T. Content services network: architecture and protocols. In *Proc. of 6th International Workshop on Web Caching and Content Distribution (IWCW6)*, 2001.

[39] Malkin, G. Traceroute using an IP option. Internet Engineering Task Force RFC 1393, 1993.

[40] Mirror Image Internet. Content Delivery and the Mirror Image Adaptive CAP Network, 2007. www.mirror-image.com

[41] Mobasher, B., Cooley, R., and Srivastava, J. Automatic personalization based on Web usage mining. *Communications of the ACM*, 43(8), pp. 142–151, 2000.

[42] Molina, B., Palau, C. E., and Esteve, M. Modeling content delivery networks and their performance. *Computer Communications*, 27(15), pp. 1401–1411, 2004.

[43] Moore, R., Prince, T. A., and Ellisman, M. Data intensive computing and digital libraries. *Communications of the ACM*, 41(11), ACM Press, NY, USA, pp. 56–62, 1998.

[44] Oram, A. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly & Associates, Inc., Sebastopol, CA, 2001.

[45] Ozsu, M. T. and Valduriez, P. *Principles of Distributed Database Systems*, Prentice-Hall, Inc., Upper Saddle River, NJ, 1999.

[46] Pai, V. S., Wang, L., Park, K. S., Pang, R., and Peterson, L. The dark side of the web: an open proxy's view. In *Proc. of the Second Workshop on Hot Topics in Networking (HotNets-II)*, Cambridge, MA, USA, 2003.

[47] Pallis, G. and Vakali, A. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1), ACM Press, NY, USA, pp. 101–106, 2006.

[48] Park, K. S. and Pai, V. S. Scale and performance in the CoBlitz large-file distribution service. In *Proc. of the 3rd Symposium on Networked Systems Design and Implementation (NSDI 2006)*, San Jose, CA, USA, 2006.

[49] Pathan, M. Content delivery networks (CDNs) research directory, 2007. http://www.gridbus.org/cdn/CDNs.html

[50] Pathan, M., Broberg, J., Bubendorfer, K., Kim, K. H., and Buyya, R. An Architecture for Virtual Organization (VO)-Based Effective Peering of Content Delivery Networks, UPGRADE-CN'07. In *Proc. of the 16th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, CA, USA, 2007.

[51] Peng, G. CDN: Content distribution network. Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, Stony Brook, NY, 2003. http://citeseer.ist.psu.edu/peng03cdn.html

[52] Pierre, G. and van Steen, M. Globule: a collaborative content delivery network. *IEEE Communications*, 44(8), 2006.

[53] Plagemann, T., Goebel, V., Mauthe, A., Mathy, L., Turletti, T., and Urvoy-Keller, G. From content distribution to content networks – issues and challenges. *Computer Communications*, 29(5), pp. 551–562, 2006.

[54] Rabinovich, M. and Spatscheck, O. *Web Caching and Replication*, Addison Wesley, USA, 2002.

[55] Rabinovich, M., Xiao, Z., Douglis, F., and Kalmanek, C. Moving edge side includes to the real edge – the clients. In *Proc. of USENIX Symposium on Internet Technologies and Systems*, Sealttle, Washington, USA, 2003.

[56] Rekhter, Y. and Li, T. A border gateway protocol 4. Internet Engineering Task Force RFC 1771, 1995.

[57] Saroiu, S., Gummadi, K. P., Dunn, R. J., Gribble, S. D., and Levy, H. M. An analysis of Internet content delivery systems. *ACM SIGOPS Operating Systems Review*, 36, pp. 315–328, 2002.

[58] Sivasubramanian, S., Pierre, G., Van Steen, M., and Alonso, G. Analysis of caching and replication strategies for Web applications. *IEEE Internet Computing*, 11(1), pp. 60–66, 2007.

[59] Szalay, A. and Gray, J. The world-wide telescope. *Science* 293(5537), pp. 2037–2040, 2001.

[60] Takase, T. and Tatsubori, M. Efficient Web services response caching by selecting optimal data representation. In *Proc. of 24th International Conference on Distributed Computing Systems*

(ICDCS 2004), pp. 188–197, 2004.
[61] Vakali, A. and Pallis, G. Content delivery networks: status and trends. *IEEE Internet Computing*, 7(6), IEEE Computer Society, pp. 68–74, 2003.
[62] Venugopal, S., Buyya, R., and Ramamohanarao, K. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Computing Surveys*, 38(1), ACM Press, NY, USA, 2006.
[63] Verma, D. C. *Content Distribution Networks: An Engineering Approach*, John Wiley & Sons, Inc., New York, USA, 2002.
[64] Wang, L., Park, K. S., Pang, R., Pai, V. S., and Peterson, L. Reliability and security in CoDeeN content distribution network. In *Proc. of the USENIX 2004 Annual Technical Conference*, Boston, MA, USA, 2004.
[65] Yim, A. and Buyya, R. Decentralized media streaming infrastructure (DeMSI): an adaptive and high-performance peer-to-peer content delivery network. *Journal of Systems Architecture*, 52(12), Elsevier, The Netherlands, pp. 737–772, 2006.

第 2 章　CDN 分类法

Mukaddim Pathan 和 Rajkumar Buyya

2.1　引言

CDN[79,97]近来引起学术上的广泛关注，研究涉及其分类、特性、缺陷、机遇以及前景。其中，Peng[75]针对现存的 CDN 给出了一个总结，介绍了在设计和实现高效 CDN 时涉及的关键问题，并对相关的解决方案进行了综述。Vakali 等人[95]针对 CDN 架构和主要 CDN 服务提供商作了综述，重点关注如何理解 CDN 框架及其效用，并给出了内容网络领域的若干特点及现有成果。同时提出了一个 CDN 的进化方向，目的在于探索现有内容网络的发展趋势。Dilley 等人[29]对 CDN 业内翘楚 Akamai[1]的整体系统架构进行了分析，提供了针对现有内容分发方法的综述，并详细描述了 Akamai 网络架构及其运行机制，指出了在构建大型 CDN（如 Akamai）时面对的技术挑战。Saroiu 等人[84]针对四大内容分发系统进行了分析：超文本传输协议（HTTP）Web 浏览、Akamai 网络、Gnutella[8, 25]和 KaZaa 点对点文件共享系统[62, 66]，并对那些大型的组织和服务提供商、网络基础设施架构提供商和一般的内容分发提供商给出了建议。Kung 等人[60]描述了 CDN 的分类并引入了一类新型的内容网络，该内容网络能够实现内容的“语义聚合和内容敏感放置”。他们基于以下两点将内容网络分为两类：内容聚合和内容放置。Sivasubramanian 等人[89]讨论了在建立一个网页副本托管系统时遇到的问题。因为缓存架构是 CDN 中的重要组成部分（如 Akamai 等），同时因为内容分发是由源服务器开始，因此他们将 CDN 视为副本托管系统。在此范围内，他们提出了一个体系框架，总结相关研究工作并进行分类。针对 P2P 内容分发技术的一个综述[11]研究了现有的 P2P 系统，并通过非功能特性如安全性、匿名性、公平性、可扩展性的增加和性能以及资源管理和组织能力等方面对其进行分类。通过该研究，作者分析了系统设计对这些特性的影响，并给出了一些有价值的结论。Cardellini 等人[20]研究了分布在局部区域内的多服务器节点网络系统架构的现状，提出了这些架构的分类法，并分析了相关路由机制及调度算法，进而提出了未来该领域的研究方向。

2.1.1　动机与范围

如上所述，现有工作已考虑了 CDN 的诸多方面，如内容分发、复制、缓存、Web 服务器放置等。然而，上述文献并未从功能性和非功能性方面对 CDN 进行分类研究。包括技术应用和 CDN 的操作，而后者关注 CDN 的特性，如组织、管理和性能等。本章的方法则考虑到 CDN 的功能性和非功能性两方面，因此有助于考察 CDN 的

架构设计决策对 CDN 特性的展现方式和影响方式。因此，本章的目标是提出一个综合的 CDN 分类，使其能够识别和分类与各种动态设计相关的技术和解决方案。

本章提出的分类法围绕建立一个 CDN 系统的核心问题展开。具体而言，本章主要关注在 CDN 构建中会带来挑战的如下问题：

（1）一个合理规划的 CDN 应该有哪些组成部分

研究内容主要包括 CDN 组织类型、节点间的交互、关系以及内容/服务类型等。

（2）如何开展有效的内容分发和管理

研究内容主要包括对于内容选择、代理放置策略、内容外包和缓存管理方法等。

（3）如何将客户端请求路由到合适的 CDN 节点

研究内容主要包括动态、可扩展和高效的路由技术。

（4）如何衡量一个 CDN 的性能

主要指预测、监控和确保 CDN 端到端性能的能力。

一个完备的 CDN 系统设计还需要考虑其他因素，以使系统更加鲁棒、容错（具有检测广域错误的能力）、安全并能够进行大范围的应用托管。在此范围内，还需要考虑如下问题：

1）如何在 CDN 中处理广域错误，即利用适当工具和系统进行错误检测。

2）如何确保广域内 CDN 系统的安全性，主要指针对分布式安全威胁的解决方案。

3）如何进行广域的应用托管，主要涉及开发合适的技术来确保 CDN 能够进行应用托管。

以上每一个问题就其自身而言都是一个独立的研究领域，其解决方案和技术均可在相关文献和具体应用中找到。虽然对于 CDN 开发而言，有必要为上面三个额外问题找到合适的解决方案，但本章研究的分类法主要关注的是首先提出的四个核心方面。然而，本章也提出了针对上述额外问题的观点，并提供了近期相关的研究工作。因此，读者能够得到足够的材料来对各方面问题进行综合，并对自己感兴趣的课题直接开展研究。

2.1.2 贡献和组织结构

本章的贡献主要分为两点：

（1）提出了一个 CDN 分类法

本章全面概括了 CDN 相关领域，并对 CDN 的应用、特性和实践技术提供了综合性的解释。该分类法的目的在于探索 CDN 的功能特性和非功能特性，并提供针对相关领域的解决方案及技术的分类。

（2）对分类法进行了验证

本章使用该分类法对若干有代表性的 CDN 进行分类和分析，以此验证其效果。此外，该分类法有助于在该领域内进行“缺口”分析，并解释相关领域内的核心概念。

本章余下内容组织如下：首先 2.2 节提出 CDN 分类法；在 2.3 节将分类法应用

到若干有代表性的 CDN 系统中，并得出若干结论；在 2.4 节中讨论 CDN 开发的其他问题，重点关注错误处理、安全和应用托管等方面；最后，在 2.5 节中给出总结。

2.2 分类法

本节给出一个 CDN 分类法，它基于四个要素，即 CDN 的构成、内容分发和管理、请求路由和性能测量，如图 2.1 所示。



图 2.1 CDN 的分类

第一个问题涉及 CDN 的组织和形成的很多方面，该分类法将 CDN 通过其结构属性进行分类；第二个问题是关于 CDN 内容分发机制，主要描述了代理放置、内容选择和分发、外包以及缓存/副本的组织等内容分发和管理方法；第三个问题描述现有 CDN 中的请求路由技术，而最后一个问题主要讨论 CDN 中衡量性能指标的方法。

2.2.1 CDN 的构成

对 CDN 结构属性的分析表明，CDN 设施中的各组件之间紧密相关。并且，CDN 结构灵活多变，随内容和服务的不同而不同。在一个 CDN 结构内，存在一组代理缓存服务器，用来生成内容分发组件。通过一些关系和机制的组合，用户请求被重定向到一个代理缓存服务器，交互协议用于 CDN 组件之间的通信。

图 2.2 展示了 CDN 基于多种结构特性的分类法。这些特性对于 CDN 组件至关重要，它们表明了 CDN 组件之间的组织、使用服务器的类型、关系、交互以及所提供的不同内容和服务。

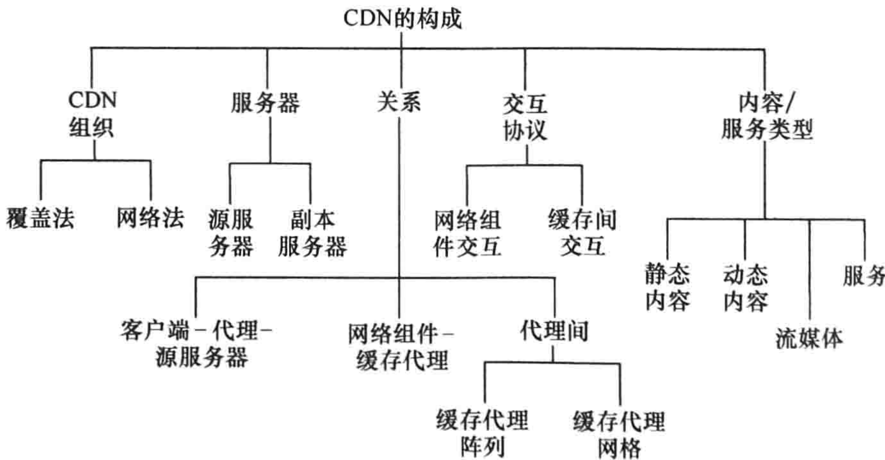


图 2.2 CDN 组构成的分类

1. CDN 组织

建立一个 CDN 一般有两种途径：覆盖法和网络法[61]。在覆盖法中，网络中不同地点的特定应用服务器和缓存服务器负责特定内容类型的分发（如网页内容、流媒

体和实时视频)。不同于针对特定请求/流量控制去提供基本网络连接和确保网络质量，核心网络组件（如路由器和交换机等）并不在内容分发中起关键作用。大多数商业CDN提供商（如Akamai和Limelight等）使用覆盖法来组织CDN。这些CDN提供商将内容复制到遍布全球的缓存服务器中，当接收到来自终端用户的内容请求时，这些请求被重定向到最近的CDN服务器上，从而加快了网站的响应时间。因为CDN提供商不需要控制内在的网络组件，因此覆盖法简化了管理并对引入新服务提供了可能性。

在网络法中，网络组件（如路由器和交换机等）中预置了相关程序，用于识别特定的应用类型以及根据预定义的策略转发请求。例如，一些设备将请求重定向到本地缓存，或者将到达数据中心的流量转到某些经优化后专用于服务特定内容类型的特定服务器。一些CDN（如Akamai、Mirror Image等）同时使用两种方法来进行组织。在这样的情况下，一个网络组件（如交换机）可以在服务器集群的前端工作，并将请求重定向到附近的特定应用代理服务器。

2. 服务器

CDN使用的服务器分为两种类型：源服务器和副本服务器。具有决定性意义和最完备的内容版本存储在源服务器上，由内容提供商负责更新。另外，一台副本服务器会存储内容的副本，但是也可以用做对客户端提供服务的授权备选服务器。源服务器与分布式的副本服务器彼此进行通信，以更新副本服务器的内容。CDN中的副本服务器可作为媒体服务器、网页服务器或缓存服务器。一个媒体服务器安装包括媒体服务器软件，可以服务于任何数字和经编码的内容。在响应特定客户端的请求时，媒体服务器可以提供特定视频或者音频片段。Web服务器包含流媒体的链接以及其他CDN需要处理的基于Web的内容。缓存服务器在网络边缘进行源服务器内容的复制（副本或内容缓存)，无须每个内容请求都经由源服务器来响应。

3. 关系

CDN复杂的分布式结构展现了组件之间不同的关系，如图2.3所示。这些关系涉及的网络组件包括客户端、代理、源服务器、代理缓存及其他网络设备。这些组件在CDN内部与副本和缓存内容进行通信。复制负责不同计算机系统上给定内容副本的创建和维护，主要是将内容从源服务器“推”到副本服务器[17]。另外，缓存完成那些可缓存响应的存储，以降低响应时间以及将来处理这些响应所需要的网络带宽消耗[26,27,99]。

在一个CDN环境中，内容分发的基本关系存在于客户端、代理缓存服务器和源服务器三者之间。客户端可以与代理缓存服务器通信，而不需要向一台或多台源服务器发送请求。在没有使用代理缓存服务器的区域，客户端则直接与源服务器通信。用户和代理缓存服务器之间的通信以透明方式进行，就如同与源服务器通信一样。代理缓存服务器通过本地缓存响应客户端请求，或充当源服务器的网关。在图2.3a中，显示了客户端、代理缓存服务器和源服务器之间的关系。

如前所述，CDN可以使用网络法构建，其中网络组件（如路由器、交换机等）通过逻辑关系部署，将流量转移到能够处理用户请求的缓存服务器（caching servers)

或代理服务器（proxy)。这种情况下，客户端、网络组件和缓存服务器/代理服务器（或代理服务器阵列）之间的关系如图 2.3b 所示。除了上述关系，CDN 中的缓存代理可以彼此之间通信。一个代理服务器中的缓存是一个应用层的网络服务，它用于缓存网络对象。代理缓存服务器可被多用户同时访问和共享。CDN 代理服务器中的缓存和 ISP 的缓存之间的关键区别在于前者仅为特定的内容提供商（即 CDN 的客户）提供内容服务，而后者服务于所有的 Web 网站。

依靠代理服务器间的通信[26]，带缓存功能的代理服务器表现为某种代理服务器阵列［见图 2.3c］或某种代理服务器网络［图 2.3d］的形式。一个缓存代理服务器阵列是缓存代理服务器之间紧密耦合的组合。在缓存代理服务器阵列中，一台代理服务器充当主服务器，与其他缓存代理服务器进行通信。一个用户代理（agent）可以与该缓存阵列进行交互。一个缓存代理服务器网络是缓存代理服务器之间某种松散耦合的组合。不同于缓存代理服务器阵列，当缓存代理服务器之间一一对应时，代理服务器网络才被创建。在缓存代理服务器网络中，通信可在同层的成员中进行，也可在分层的子成员和父成员间中进行。一台缓存服务器在代理服务器网络中充当网关，转发来自客户端处的本地代理服务器所发出的请求。

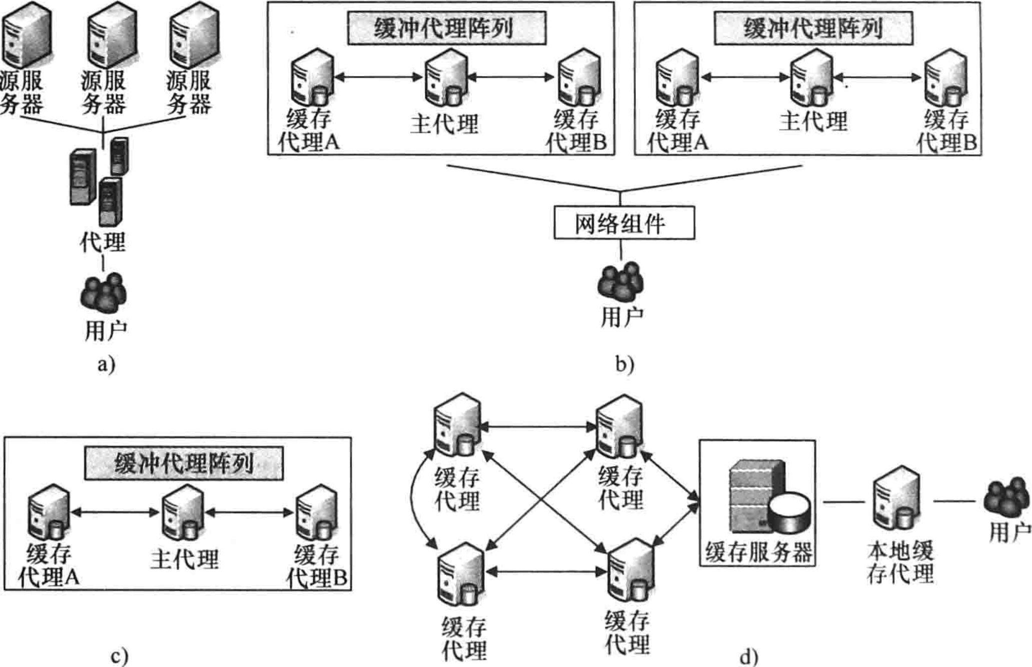


图 2.3 关系

a）客户端—代理缓存服务器—源服务器 b）网络中组件到缓存代理服务器

c）缓存代理服务器阵列 d）缓存代理服务器网络

4. 交互协议

基于前述的通信关系，现在可以辨识 CDN 组件之间使用的交互协议。交互可以大致分为两类：网络组件间的交互和缓存间的交互。图 2.4 展示了一个 CDN 内组件之间交互所使用的不同协议。网络组件之间交互协议包括网络组件控制协议（NECP）和网页缓存控制协议（WCCP），而缓存交互协议包括缓存阵列路由协议

(CARP)、互联网缓存协议（ICP)、超文本缓存协议（HTCP）以及缓存摘要（Cache Digest）等。



图2.4　交互协议

网络组件控制协议（NECP)[24]是服务器和转发流量的网络组件之间彼此通信时使用的轻量级协议。网络组件由若干设备组成，包括“内容已知”的交换机和负载均衡路由器，NECP允许网络组件在服务器集群间保持负载均衡以及完成至监听代理服务器的重定向。然而，它并不规定任何特定的负载均衡策略，相反NECP向网络组件提供感知服务器容量和有效性的方法，并告知哪些请求可以被服务而哪些不行。因此，网络组件收集必要信息来做出负载均衡的决策，从而避免了协议使用的专用性带来的协议间互不兼容问题。NECP可以广泛用于各种服务器应用中，包括源服务器、代理服务器和监听代理服务器。NECP使用传输控制协议（TCP)，当服务器初始化时使用标准端口在网络组件间建立一个TCP连接。消息可以在服务器和网络组件间双向传递。大多数消息包括一个带有反馈和确认的请求。接收到肯定性确认信息意味着一台peer的若干状态得到了记录，这个状态可假定一直存在于peer中直到过期或者peer停止运转。换言之，该协议使用了“硬状态”模型。应用层的KEEP ALIVE消息被用于检测在此通信情况中的崩溃状态。当一个节点检测到它的peer已经崩溃，它就假设该peer中的所有状态均需要在peer启动后重新加载。

网页缓存控制协议（WCCP)[24]负责一个或多个路由、一个或多个网页缓存之间的交互。它工作在监听代理服务器与充当重定向网络组件的路由器之间。这样交互的目的在于建立和维护一组路由器中被选择流量类型的透明重定向。被选择的流量被重定向到一组网页缓存中，用来增强资源利用率和最小化响应时间。WCCP允许一台或多台代理服务器注册单一路由来接受重定向流量。流量包括用户浏览WWW服务器中页面和图片的请求（无论在网络内部还是外部)，并响应这些请求。该协议允许指定其中一台代理服务器来告知路由器：重定向流量如何在代理服务器阵列中分布。WCCP不仅提供技术手段来具体协调用于分配网页缓存间负载的特定方法，还提供了路由和缓存之间传输流量的方法。

缓存阵列路由协议（CARP)[96]是一个基于松散耦合代理服务器列表和一个用于划分这些代理服务器之间URL空间的哈希函数的分布式缓存协议。一个执行CARP的HTTP客户端实现的CARP可以将用户请求路由到代理服务器阵列中的任一成员。代理阵列成员表可以通过阵列配置URL获得的简明ASCII文本文件定义。CARP的哈希函数和路由算法提取代理服务器阵列成员表中一个预先定义的成员，并动态决定哪

一个成员应当成为被 URL 指向的资源缓存版本的适当容器。因为请求是通过代理服务器进行排序的，因此缓存内容的副本将被删除，同时全局缓存命中率也得以提高。下级代理服务器可以通过将被代理的 HTTP 资源请求转发给合适的代理服务器阵列成员来访问缓存资源。

互联网缓存协议（ICP）[101]是一个轻量级的报文格式，用于缓存间的通信。缓存交换 ICP 请求并响应、收集报文，用于选择最合适的地点来定位目标。不同于作为目标定位协议，ICP 报文可同样应用于缓存选择。ICP 是一个广泛应用的协议。虽然网页缓存使用 HTTP 实现目标数据的传输，但大多数缓存代理服务器都以某种形式支持 ICP。ICP 被用在缓存代理服务器网络中，可以实现相邻缓存中特定网页目标的定位。一个缓存服务器发送 ICP 请求到它的相邻服务器，同时相邻服务器的响应指示出是否“命中”。若短时间内未能收到相邻服务器的回复，则表明网络阻塞或者断网。一般而言，ICP 工作在用户数据报协议（UDP）之上用以提供网页缓存应用的重要特征。因为 UDP 是不可靠和无连接的网络传输协议，因此可以通过计算 ICP 丢失率来估计网络的阻塞和可用性。这种丢失率检测和往返时间（RTT）一起，提供了一种缓存之间负载均衡的方式。

超文本缓存协议（HTCP）[98]用于寻找 HTTP 缓存、被缓存数据和 HTTP 缓存管理集，并监控缓存特性。HTCP 和 HTTP1.0 兼容，这一点与 ICP 不同，因为 ICP 是针对 HTTP 0.9 版本推出的。HTCP 也扩展了缓存管理域，使其包含监控远程缓存的添加和删除、请求立即删除和发送关于网页目标的建议，如可缓存目标的第三方定位，或者关于测量得到的不可缓存特性或者网页目标不可获得性的建议。HTCP 报文可通过 UDP 或者 TCP 发送。HTCP 代理必须与网络错误和延时结合在一起。一个 HTCP 代理应当在服务器没有响应或者响应信息被丢失或损坏的情况下充当有用角色。

缓存摘要[42]是一个交换协议和数据形式。它提供了响应时间和其他缓存间通信协议(如 ICP 和 HTCP 等)所导致出现的拥塞问题的相关解决方案。它能够在请求—响应交换没有发生的情况下，支持缓存服务器之间的对等，相比而言，其他与其对等的服务器都需要取得关于服务器内容的汇总（如摘要等）。当使用缓存摘要时可以精确判断某一特定服务器是否缓存一个给定的 URL。通常的做法是通过 HTTP 实现。一个对等成员在收到对其摘要的请求后，响应将通过 HTTP 失效头信息（HTTP Expires header）来确定摘要失效时间。因此，请求缓存能够知道它何时应当请求对等摘要的最新副本。除了 HTTP 之外，缓存摘要也可以通过 FTP 进行交换。虽然缓存摘要主要用于对给定服务器 URL 的缓存汇总的共享，但是它也可以被扩展用来覆盖其他数据源。缓存摘要在用来消除冗余和更好地利用互联网服务器和带宽资源时可以作为非常有用的机制。

5. 内容/服务类型

CDN 提供商对第三方内容进行托管，实现任何数据内容的快速分发，包括静态内容、动态内容、流媒体（如音频和实时视频），以及不同内容服务（如目录服务、电子商务服务和文件传输服务等）。这些内容来自大企业、Web 服务提供商、媒体公司以及新闻媒介。内容和分发服务的不同需要一个 CDN 提供商选择使用特定的特性、

架构和技术。出于这个原因，一些 CDN 专注于特定内容分发和（或）服务。下面将分析内容和服务类型的特性以展示它们的同质性。

静态内容指那些改变频率很慢的内容，它们不随用户请求而变化。静态内容包括静态 HTML 页面、内嵌图片、可执行文件、PDF 文档和软件包、音频和（或）视频文件等。所有的 CDN 提供商都支持这一类型的内容分发。此类型的内容便于缓存，其更新可以通过传统缓存技术来维护。

动态内容指那些基于用户个性化需求的内容或通过执行某些应用进程得到的基于需求的内容。这些动态内容变化迅速，通常取决于用户需求，包括动画、脚本以及 DHTML。由于其频繁变化的本性，一般被认为不可缓存。

流媒体可以是直播的也可以是点播的。直播媒体的分发可以用于直播节目，如体育节目、音乐会、频道和（或）新闻广播。在这种情况下，内容通过编码器实时发送到媒体服务器，最终传送到客户端。在点播分发中，内容编码完成后，以流媒体文件存储在媒体服务器中，内容可基于客户端的请求而获得点播的媒体内容包括用户点播的音频、视频、电影文件和音乐剪辑。流服务器使用专用协议通过 IP 网进行内容的分发和更新。

一个 CDN 可以提供自身的网络资源作为服务分发通道，并允许增值服务提供商以互联网基础设施服务的方式进行应用服务。当边缘服务器托管增值服务软件以供内容分发时，它们的运行机制与转码代理服务器、远程调出服务器或代理服务器很类似[64]。这些服务器也验证了增值互联网架构服务的处理和特殊托管的可行性。CDN 提供的服务可以是目录、网络存储、文件传输和电子商务服务等。CDN 提供的目录服务被用来访问数据库服务器。用户对特定数据的请求被转发到数据库服务器，同时那些经常访问的结果被缓存在 CDN 的边缘服务器上。网络存储服务是指在边缘服务器上存储内容，且同样是基于静态内容分发技术。文件传输服务方便了软件的分发、病毒库升级、电影点播以及高清晰度医学影像在全球范围内的分发。所有这些内容在本质上均为静态内容。网络服务技术通过 CDN 的维护和分发来进行更替。电子商务是一种非常流行的通过 Web 进行商业交易的手段。电子商务服务中的购物车可以在边缘服务器上进行存储并维护，同时在线交易（如第三方认证、信用卡交易等）也可以通过边缘服务器进行。为了方便此项服务，CDN 边缘服务器应当具备对于电子商务站点动态内容的缓存能力。

2.2.2　内容分发和管理

内容分发和管理对于 CDN 的高效内容分发和整体性能而言至关重要。内容分发包括：

（1）内容选择和分发

内容选择和分发基于特定用户的请求频率和请求类型。

（2）代理缓存服务器的放置

代理缓存服务器按照某种特定的策略放置，使边缘服务器在地理位置上贴近用户。

(3) 内容外包

决定采取哪一种外包策略。

(4) 内容管理

内容管理很大程度上取决于缓存组织，如缓存技术、缓存维护和缓存更新。内容分发和管理的分类如图 2.5 所示。

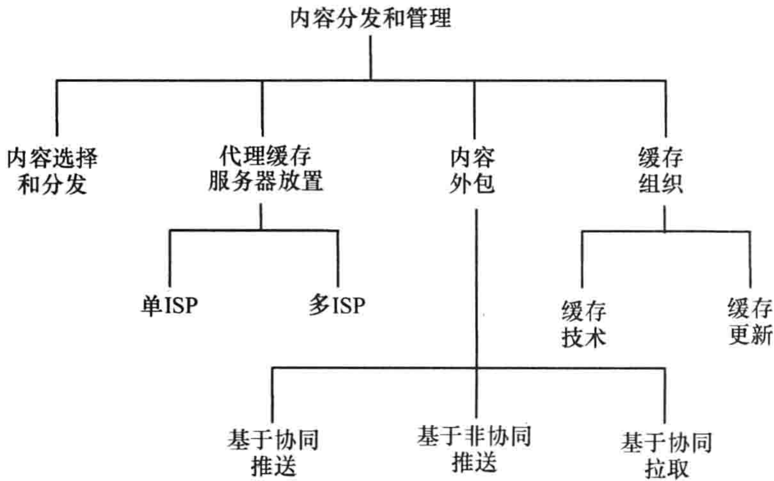


图 2.5 内容分发和管理的分类

1. 内容选择和分发

内容分发的效率高低在于恰当地选择分发至终端用户的内容。一个合适的内容选择方法可以有助于减少客户端的下载时间和服务器负载。图 2.6 显示了内容选择和分发技术的分类。从图中可以看到，内容可以全部或部分地发送给用户。

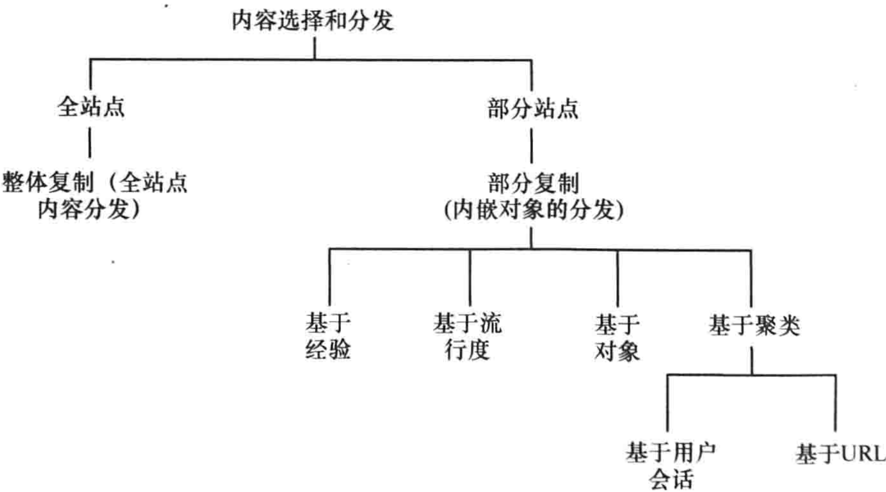


图 2.6 内容选择和分发技术的分类

将整个网站的内容发送出去是一种简单的方法，这个过程中代理缓存服务器执行“整体复制”功能，将所有内容全部分发给终端用户。通过这样的方式，所有的客户端请求仅通过一个 CDN 服务器进行处理，该 CDN 服务器负责发送全部内容，而内容提供商则据此来配置自己的 DNS。该方法的好处在于简单，然而考虑到网络对象在持

续增长，该方法并不适用。尽管存储硬件的价格在下降，但边缘服务器的存储空间永远达不到存储所有内容的要求。另外，因为网站内容并非静态，因此更新如此巨大的 Web 对象集是不可行的。

同时，在选择和分发部分内容的方法中，代理服务器执行“部分复制”功能，仅从相应的 CDN 服务器中分发某些内嵌对象，如网页中的图片。通过部分内容分发的方式，内容提供商对内容进行修改，使得指向特定对象的链接在经 CDN 服务商授权过的域中具有主机名。因此，网站的主页从源服务器获取，同时网页中的内嵌对象通过 CDN 缓存服务器获取。部分选择的方法要比全部选择的方法更好：前者既减轻了源服务器的负载，也缓解了负责创建内容的网络设施。此外，因为内嵌内容并不需要频繁更新，故部分选择的方法具有更好的性能。

内容选择取决于合适的用于复制网络内容的管理策略。基于选择内嵌对象来进行复制的方法中，部分复制方法可进一步被分为基于经验的、基于访问量的、基于对象的和基于聚类的内容复制。在基于经验的方法中[23]，网站管理员凭经验选择需要复制到边缘服务器的内容。实现这一经验性决策通常使用启发式方法。该方法的缺陷在于选择正确启发式方法时的不确定性。在基于访问量的方法中，访问量最高的对象被复制到代理服务器中，这种方法很耗时而且无法保证获得可靠的对象请求统计信息，因为每一对象的访问度差别极大。此外，这种统计方法无法用于新加入的内容。在基于对象的方法中，内容以对象为单位复制到代理服务器中。该方法属于贪婪法，因为每一个对象被复制到能使性能增益最大的代理服务器中（在存储容量的限制条件下）[23,102]。虽然贪婪法能够达到最好的效果，然而在具体应用中其实现复杂度相当高。在基于聚类的方法中，网络内容通过相关性或者访问频度等指标进行聚类，并以不同的分类为单位进行复制。因为无论聚类数目还是聚类的大小都是未知的，所以聚类处理都是以固定的聚类个数或者固定的最大聚类大小来进行。内容聚类法可以基于用户会话或基于 URL。在基于用户会话的方法中[36]，网页日志文件按照用户网络会话的相似性进行用户的聚类。这种方法的优点是有助于判定具有相同浏览模式的用户组和相关内容的网页。在基于 URL 的方法中，网络内容的聚类通过网站拓扑决定[23,26]，最热门的对象通过网站确定并以聚类单位进行复制，其中每一对 URL 之间的相关距离是基于特定的相关度量计算得到。实验数据表明，基于聚类的方法降低了用户的下载时间和服务器的负载。然而，此类方法的缺点在于部署的复杂度。

2. 代理缓存服务器的放置

因为代理缓存服务器（surrogate server）的放置位置与内容分发过程紧密相关，所以为每一台代理缓存服务器选择最佳放置地点至关重要。最优放置的目的在于降低用户访问内容时的延时感受以及最小化从服务器到客户端传输复制内容的全局网络带宽消耗。以上两种度量的优化可以降低 CDN 提供商的设施和通信成本。因此，代理缓存服务器的最优放置使 CDN 能够提供高质量和低成本的服务[88]。

图 2.7 展示了不同的代理缓存服务器放置策略。最小 k – 中心问题和 k – 层结构分离树（K – HST）等理论方法将服务器放置问题建模为中心放置问题，其定义为：对于给定中心放置的数量，使一个节点与最近中心的最大距离最小化。K – HST 算

法[16,47]通过图论来解决服务器放置问题，其中网络由图 $G(V, E)$ 表示，其中 V 是节点集，$E \subseteq V \times V$ 是链路集。算法由如下两步组成。

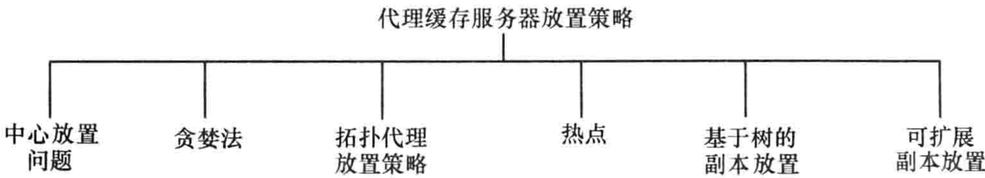


图 2.7 代理缓存服务器的放置策略

第一步，从完全图（父分类）中任选一个节点，该节点某邻域内的所有节点组成一个新的分类（子分类）。子分类的半径是 k，其中 k 小于父分类的直径。持续进行该过程，直到所有节点都位于自己的分类中。然后，经迭代得到一个分类树，根节点是整个网络，叶节点是网络的每一个节点。

第二步，一个虚拟节点被分配到每一层的每一个分类上。每一个父分类上的虚拟节点成为子分类上的虚拟节点的父节点。每一个虚拟节点汇聚起来形成树。随后使用贪婪策略，找到 K－HST 树需要的中心数量，其中中心与节点的最大距离不大于 D。

最小 k－中心问题[47]如下所述：

1）对于给定的图 G（V, E），其每个边都按照权重 c 的非递减次序排列：$c(e_1) \leqslant c(e_2) \leqslant \cdots \leqslant c(e_m)$，构造一组平方图 G_1^2、G_2^2、G_m^2。G 的每一个平方图用 G^2 表示，包含若干节点 V 和边（u, v）。

2）计算每一个 G_i^2 的最大独立集 M_i。一个 G^2 的独立集是指 G 的节点集，G 中的节点之间至少存在三跳，一个最大独立集 M 定义为独立集 V' 使得图 $V - V'$ 中的所有节点都与 V' 中的节点相距最多一跳。

3）找到最小的 i，使得 $M_i < K$，将其用 j 表示。

4）最终，M_j 为具有 K 中心的集。

由于这些算法的计算复杂度较高，所以提出了一些启发性算法如贪婪副本放置和拓扑已知放置策略。这些算法属于次优算法，利用了 CDN 中，如工作负载模式和网络拓扑等已有信息，提供了在低计算成本下的解决方案。例如，贪婪算法[59]在 N 个候选节点中选择 M 个服务器。在第一轮，计算每一个节点的开销。假设对所有客户端的访问最终收敛到某个节点。算法将选择具有最低开销的站点。在第二轮，搜索与上次找到的站点相连的具有次优开销的下一个站点。重复该操作，直到 M 个服务器完成排序。贪婪算法能够在不完备的输入数据中正常工作，但是它需要网络中客户端位置以及所有节点间距的先验知识。在拓扑放置策略[48]中，服务器被放置在以出度（即其他节点和该节点相连的个数）降序排列的备选主机上。此处假设具有更多出度的节点能够以较小的延时与更多节点通信。此方法使用自治域（AS）拓扑，其中每一个节点都代表一个单一的自治域，节点链接对应边界网关协议（BGP）的对等。在一个改进后的拓扑放置策略中[81]，使用了路由层的网络拓扑而非自治层的拓扑。在此方法中，每一个与路由器相连的 LAN 都是一个潜在站点，用于放置一台服务器，而非每一个 AS 是一个站点。

其他的服务器放置算法（如热点法（Hot Spot）[78]和基于树的副本放置方法（Tree－based）[63]）也可以用于这种情况。热点法将副本放置在产生负载最大的客户端附近。将 N 个潜在站点根据其周围产生的流量进行排序，并将副本置于前 M 个产生最大流量的站点处。基于树的副本放置方法的原理是基于某种假设，即潜在的拓扑结构是树。该算法将副本放置问题建模为动态规划问题，在此方法中，一个树 T 被划分为若干小树 T_i，同时 t 个代理的放置通过在每一个小树 T_i 中以最优方式置放 t'_i 个代理的方法来实现，其中 $t = \sum_i t'_i$。另外一个算法是扫描法[21]，是一种可拓展的副本管理框架，能够产生按需副本并将其组织成为应用层上的组播树。此方法能在满足客户延时约束和服务器容量限制的前提下最小化副本服务器数量。更多关于扫描法的内容可参阅本书第 3 章。

对于代理缓存服务器的放置，CDN 管理者同样需要决定服务器的最优数量，一般采用单 ISP 法或多 ISP 法[95]。在单 ISP 法中，一个 CDN 提供商一般在网络边缘部署至少 40 台代理缓存服务器来支持内容分发[30]。单 ISP 法的策略，是在每一个主要城市的 ISP 网络内放置一到两台代理缓存服务器，并为代理缓存服务器配置较大的缓存能力。因此，一个全球网络的 ISP 可以具有极大的地理覆盖能力，并不需要依赖其他 ISP。单 ISP 法的缺点在于代理缓存服务器有可能被放置在离 CDN 提供商的用户较远的地点。在多 ISP 法中，CDN 提供商尽可能多地在全球 ISP 的网络接入点（POP），部署代理缓存服务器。这样，可以解决单 ISP 面临的问题，将代理缓存服务器放置在离用户很近的地点，使内容可以可靠、及时地分发给用户。大型 CDN 提供商如 Akamai 拥有超过 25000 台服务器[1,29]。但是除了成本和建设复杂度之外，多 ISP 法的主要缺点在于，每一台代理缓存服务器接收到的内容请求变得更少（甚至没有），这会导致资源的闲置和 CDN 性能的低下[71]。对以上两种方法的性能估计表明，单 ISP 法对于低到中等流量的站点表现良好，而多 ISP 法适用于大流量的站点[30]。

3. 内容外包

在 CDN 基础架构中给定一组放置恰当的代理缓存服务器和选定需要分发的内容之后，选择高效的内容外包方法至关重要。内容外包可以使用基于协作推送法、基于非协作拉取法和基于协作拉取法等。

基于协作推送的方法需要对被送到代理缓存服务器的内容进行预读取。内容从源服务器中被推送到代理缓存服务器，代理缓存服务器之间通过彼此协作来降低复制和更新的成本。在此方案下，CDN 维护内容与代理缓存服务器之间的映射关系。每一个请求都被指向最近的代理缓存服务器，或者直接指向源服务器。在这种方法中，使用全局贪婪启发式算法，该算法适合在协作代理缓存服务器之间做出复制决策[54]。然而，由于该方法尚未被任何商业 CDN 提供商使用，所以目前只在理论上具有可行性[23,36]。

在基于非协作拉取的方法中，客户端请求被转发到距离他们最近的代理缓存服务器中，如果出现缓存未命中，代理缓存服务器会从源服务器中读取数据。大多数主流 CDN 提供商（Akamai、Mirror Image 等）都使用这种方法。该方法的缺陷在于很难选

定最优的服务器来服务内容请求[49]，多数 CDN 使用这种方法的原因是基于协作推送的方法仍然处于试验阶段[71]。

协作拉取方法和非协作拉取方法的区别在于，如果缓存未命中，则前者是通过代理缓存服务器之间的彼此协作得到请求内容。在协作拉取方法中，客户端请求通过 DNS 重定向机制被转到最近的代理缓存服务器。通过使用分布式索引，代理缓存服务器能够在附近找到请求内容的副本，并在缓存中对其进行存储。协作拉取方法是反应式的，只有客户端请求某数据时该数据才被缓存。一个学术 CDN（Coral[34]）就是使用分布式索引和协作拉取方法，在缓存未命中的情况下进行代理缓存服务器之间的协作。

对内容外包来说，决定对哪一台代理缓存服务器上的外包内容进行复制至关重要。若干参考文献中显示了对外包内容使用不同复制策略的效果。Kangasharju 等人[54]使用了四种启发式方法，即随机法、流行度法、单一贪婪法和全局贪婪法，用于外包内容的复制。Tse[94]提出了一系列贪婪法，通过均衡代理缓存服务器的负载和容量来确定其放置。Pallis 等人[72]提出了一种自调整和无参数的算法，称为 lat - cdn，该算法使用对象延时来做出复制决策，实现 CDN 代理缓存服务器外包内容的最优放置。一个对象的延时被定义为从发出对网络对象的请求到完整收到该对象之间的延时。lat - cdn 的一个改进是 il2p[70]，其原理是把延时和对象负载一并考虑，实现外包对象到代理服务器的放置。

4. 缓存组织和管理

内容管理对于 CDN 性能而言至关重要，主要与缓存组织密切相关。缓存组织不仅涉及缓存技术和缓存更新频率（确保内容的时效性、可获得性和可靠性），还包括缓存和复制的综合利用，这对 CDN 的高效内容管理很有用。如果在 CDN 中同时使用复制和缓存，那么感知延时、命中率和字节命中率等方面的性能都可得到提升[91]。另外，复制和缓存的组合能够有助于应对瞬时拥塞问题。在此意义下，Stamos 等人[90]提出了泛化的非参数启发式方法，能够将 Web 缓存和内容复制集成起来。他们提出了一种放置相似性的方法，称为 SRC，用于对集成的水平进行评估。另一种集成方法称为 Hybrid，将静态复制和 Web 缓存通过一种 LRU 模型（由 Bakiras 等人[13]提出）集成起来。Hybrid 法能够迭代地用静态内容逐渐填充代理缓存服务器的缓存，只要该内容有助于改善响应时间。更多关于缓存和复制集成使用的内容可参阅本书第 4 和第 5 章。

CDN 中的内容缓存可以采用类内机制或类间机制。一个缓存技术的分类法如图 2.8 所示。基于查询、基于摘要、基于目录和基于哈希的体系方法都可以用于类内的内容缓存。在基于查询的方法中[101]，如果缓存未命中，那么 CDN 服务器将广播一个请求给其他协作 CDN 服务器。这种方法面临的问题是查询流量极大和延时问题，因为发出广播的 CDN 服务器需要在没有一个对等成员告知拥有被请求的内容之前，必须等待来自协作的所有对等服务器的响应，直到收到最后一个“未命中”信号为止。因为这些缺点，基于查询的方法受制于使用过程中产生的开销。基于摘要的方法[83]解决了在基于查询法中存在的海量查询请求问题。在该方法中，每一个 CDN 服务器

维护一个由其他协作代理缓存服务器保持的内容摘要。协作代理缓存服务器通过更新 CDN 服务器来得知任何更新的内容次序。通过检查内容摘要，一台 CDN 服务器可以将一个内容请求路由到特定代理缓存服务器。为了确保正确的信息同步，相互协作的代理缓存服务器之间需要频繁地进行信息交换，这就产生了大量的通信流量，所以这种方法的缺点是更新所带来的流量造成了较大的系统开销。基于目录的方法[38]是一个基于摘要法的中心化版本，其中一个中心化的服务器保持一个类内所有协作代理缓存服务器的所有内容，每当进行本地更新时，每台 CDN 服务器仅仅通知目录服务器，并查询目录服务器是否存在本地缓存中内容缺失的情况。这种方法存在潜在的性能瓶颈和单点失效问题，因为目录服务器是从所有协作代理缓存服务器中收到更新和查询流量的。在基于哈希的方法中[55, 96]，所有的协作 CDN 服务器维护同一个哈希函数，一个指定的 CDN 服务器保持着基于内容的 URL、CDN 服务器的 IP 地址以及哈希函数的相关内容。所有针对该特定内容的请求都被指向被指定的服务器。基于哈希的方法较其他方法而言更为高效，因为该方法具有最小的系统开销和最高的内容共享效率。然而，它对本地请求和多媒体内容的分发效率不佳，因为本地用户的请求被转到其他被指定的 CDN 服务器进行服务。为了解决这个问题，可以使用某种半哈希方法[24, 67]。在半哈希法中，本地 CDN 服务器分配一部分磁盘空间用于缓存大多数本地用户所需要的常见内容，其余磁盘空间用于与其他的 CDN 服务器通过哈希函数进行协作。如同纯哈希法一样，半哈希法也具有低系统开销和高内容共享效率的特点，同时，它还能明显增加 CDN 的本地命中率。



图 2.8　缓存技术的分类

基于哈希的方法并不适用于类间协作缓存，因为不同类所代表的 CDN 服务器在地理上呈正态分布。摘要法和目录法也并不适合类间协作缓存，因为那些代表性的 CDN 服务器需要维护一个很大的内容摘要和（或）目录，以记录其他类中的内容信息。因此，基于查询的方法可以用于类间缓存[68]。在这个方法中，当一个类未能对用户内容请求服务时，它会查询相邻的类，如果内容可以通过相邻类得到，该类就返回“匹配”报文；否则，它将请求转发给其他相邻类。一个类内的所有 CDN 服务器均使用哈希法进行内容请求服务，而类中那个代表性的 CDN 服务器仅仅向指定的服务器进行查询，从而实现对内容请求的服务。因此，这个方法使用哈希法进行类内的内容路由，并使用基于查询的方法实现类间的内容路由。因为限制了查询流量的规模和解决了从远端服务器通过每一个检索报文的空闲时间和 TTL 值去检索信息的延时问题，所以该方法在性能上有所提高。

CDN 中代理缓存服务器上的被缓存对象具有相关的失效期，一旦过期即被认为失效。确保内容的时效性对于客户端服务的信息更新而言至关重要。如果在传递信息中存在延时，CDN 提供商就需要注意内容是否一致和（或）过期。为了管理副本服务器上内容的一致性和时效性，CDN 应用了不同的缓存更新技术。缓存更新机制的分类如图 2.9 所示。

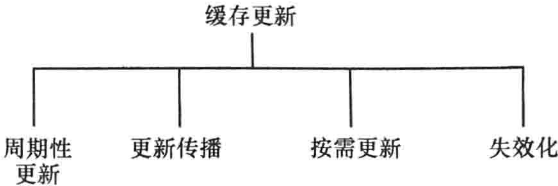


图 2.9 缓存更新机制的分类

最常见的缓存更新机制是周期性更新。为了确保内容的一致性和时效性，内容提供商在源 Web 服务器上进行功能配置，告知缓存什么内容是可缓存的、不同内容的时效性为多久以及何时去检索源服务器上的内容更新等[41]。通过这样的方式，缓存以某种固定的方式刷新。但是，这种方法的缺点在于每次更新间隔都会产生大量不必要的流量。更新传播机制在内容发生改变时被激活，它主动地将内容推送至 CDN 缓存服务器。通过这种机制，只要源服务器上的内容有所改变，一个文档的更新版可以传播至所有缓存。对于频繁改变的内容，这个方法会产生较大的更新流量。按需更新是另一种缓存更新机制，基于对内容的请求，某文档的最新副本被传递至代理缓存服务器。该方法假定：除非有特定相关请求，否则结构和内容无须更新。该方法的缺点是，为了确保被传递的内容是最新的，源服务器和缓存之间存在大量的往返流量。另外一种缓存更新技术称为失效法，每当一个文档在源服务器上发生变化，一个失效报文就被送往所有代理缓存服务器的缓存。在文件改变时代理缓存服务器被暂时禁止访问该文档，每一个缓存需要稍后自行获取文档的更新版。该方法的缺陷在于未能充分利用内容分发网络的分布式机制，而且延后的内容获取可能会导致内容一致性管理的低效。

一般而言，CDN 给内容提供商以控制内容时效性的能力，并确保内容在所有的 CDN 节点是一致的。然而，内容提供商可以自行设立策略和使用一些启发式方法来部署与其组织形式相适应的缓存策略。第一种情况下，内容提供商以一种独特的方式确定他们给 CDN 提供商的缓存策略，该方式下能将规则集传递给相关缓存，这些规则告知缓存服务器如何通过一致性来维持内容的时效性。第二种情况下，内容提供商可以应用一些启发性方法而不是开发复杂的缓存策略。通过这种机制，某些缓存服务器可以自适应地通过源服务器中内容的更改频率进行自学习，并相应调节自己的运行方式。

2.2.3 请求路由

一个请求路由系统负责将客户端的请求路由到合适的代理服务器，它包括一系列网络组件来支持单一 CDN 的请求路由，并将用户请求指向到距离其最近的副本服务器上。然而，离用户最近的服务器未必是对用户请求响应最快的代理缓存服务器[22]。因此，一个请求路由系统使用一组度量指标如网络邻近性、用户感知延时性、距离和副本服务器负载等，将用户请求转到最近的能提供最好服务的代理缓存服务器上。内容选择和分发技术（如整体复制法和部分复制法）都可以对 CDN 的请求路由系统产生影响。如果 CDN 使用整体复制法，那么请求路由系统有助于将用户请求指向代理缓存服务器，因为它们持有所有的外包内容。另外，如果采用部分复制法，那么请求路由系统会以如下方式进行设计：在接收到用户请求后，源服务器传递基本内容，同时代理缓存服务器传递内嵌对象。CDN 中的请求路由系统由两部分组成：请求路由算

法的部署和请求路由机制的使用[89]。一个请求路由算法在收到用户请求后被启动，它确定如何选择边缘服务器用于响应特定的用户请求。另外，一个请求路由机制是一种通知用户选择的方法。该机制首先启动请求路由算法，随后将它的选择结果通知用户。

图2.10提供了一个典型CDN环境中请求路由系统的高层示意图。交互流程为：（1）用户通过浏览器输入URL，向内容提供商请求内容，用户的请求被指向源服务器；（2）当源服务器收到请求后，它只提供基本内容（如网站的主页）；（3）对于需要高速宽带和频繁检索的内容（如内嵌对象——最新内容、导航条、广告条等），内容提供商的源服务器将用户请求重定向至CDN提供商；（4）通过恰当的选择算法，CDN提供商选择离用户最近的副本服务器，用于提供被请求的内嵌对象；（5）被选定的副本服务器通过源服务器得到内嵌对象，响应用户请求，并缓存内容以供下次使用。

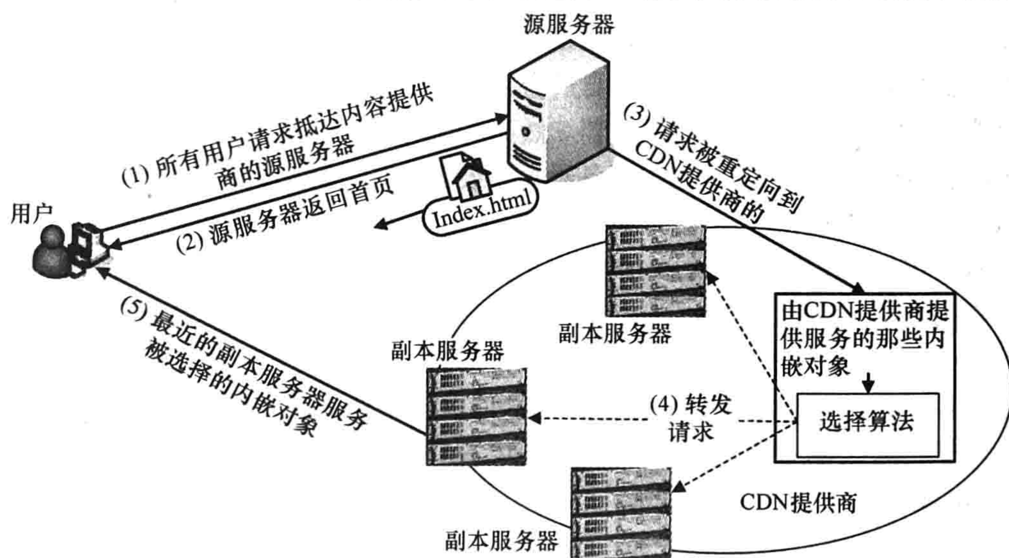


图2.10　CDN环境中的请求路由

1. 请求路由算法

请求路由机制使用的算法可以是自适应的也可以是非自适应的（见图2.11）。在进行内容分发时，自适应算法根据当前的系统状况来确定缓存服务器。系统当前状态可以来自对某些信息的估计，如副本服务器上的负载或选定网络连接的拥塞等。非自适应的请求路由算法使用某些启发式方法来选择缓存服务器，而不考虑当前的系统状态。非自适应算法容易实现，而自适应算法则较为复杂。自适应算法的复杂度在于，该算法需要改变自身行为去配合某种持续状态。非自适应性算法在启发式条件满足的情况下运行效率较高。另外，自适应算法在面对瞬时拥塞时具有较高的系统鲁棒性[100]。

一个最常见的非自适应性请求路由算法的简单例子是轮询算法，它将所有的请求分配给CDN服务器并试图平衡其负载[93]。该算法假定，所有缓存服务器都具有同样的处理能力，并且任意一台都能服务客户请求。这样的简单算法对于所有副本服务器都聚在一起时非常有效[69]。但是，该算法在缓存服务器呈现松散分布的大范围分布式系统中性能一般。因为在这种情况下，算法并不考虑副本服务器间的距离。因此，客户端请求可能会被转发到距离更远的副本服务器，从而导致算法性能下降。另外，负载均衡的目标也无法达到，因为处理不同请求需要完全不同的计算成本。

在另一个非自适应的请求路由算法中，所有副本服务器根据加载在自身上的负载的预测值进行排序。该预测可以通过迄今为止该服务器所服务过的请求数量进行估计。该算法考虑到了客户端和服务器之间的距离，同时客户端请求以均衡负载的方式被转发到副本服务器，因为算法假定副本服务器负载与客户端和服务器间的距离是对用户请求处理效果影响最重要的因素[89]。虽然 Aggarwal 等人[9]已经发现采用这种算法对于请求路由表现良好，但是客户感知到的系统性能仍然不高。

还有一些令人感兴趣的非自适应请求路由算法在思科公司的 Distributed Director 中已得到应用[28]，其中一个算法考虑了每个副本服务器接收到的用户请求的百分比，其思路是认为接收到更多请求的服务器的性能是更好的。因此，客户端请求被转向性能更好的服务器，以期得到更好的资源利用效率。另一种算法定义了服务器之间的优先级，指派那些优先级高的服务器去服务用户请求。DistributedDirector 也支持随机地将请求分配到副本服务器。并且，某些其他非自适应性算法也考虑到了用户的地理分布，并能够将用户请求重定向到附近的副本服务器。然而，这些算法表现并不理想，原因在于客户端请求可能会被分配到过载的副本服务器，导致客户端性能体验的恶化。

Karger 等人[55]提出了一个请求路由算法以自适应地应对热点（hotspot）问题。该算法基于内容的 URL 从一个大的标识符空间中计算一个哈希函数 h。这个哈希函数被用于将用户请求高效地路由到一个包含有同一空间范围内所有缓存服务器（也就是说其 ID 属于同一空间）的逻辑环。该算法规定由那个大于 h 值的最小 ID 号的缓存服务器负责保存被请求的数据，并将客户端请求定向到该服务器。该算法的一些修改版本已用于类间缓存内容[67, 68]和 P2P 文件共享[14]。

在 Globule[76]使用的一种自适应请求路由算法中，通过网络邻近度来选择距离客户端最近的服务器[93]，该度量基于周期性更新的路径长度。Globule 使用的度量估计服务是被动式的，并不涉及其他网络流量。然而，Huffaker 等人[45]的研究表明，距离度量估计的过程并不精确。

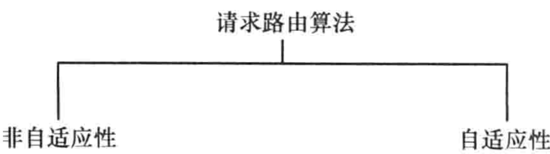


图 2.11　请求路由算法的分类

Andrews 等人[10]和 Ardiaz 等人[12]提出了一种基于客户端—服务器延时的自适应请求路由算法。在此方法中，算法通过客户端访问日志或被动的服务器端延时测量，来决定客户端请求需要发送给哪一台副本服务器。这样，客户端的请求可以转发到具有最小延时的副本服务器上。这个算法非常有效，因为它考虑到延时因素。然而，这种实现方式需要维护一个中心数据库来管理测量数据，因而会限制使用该算法的系统的可拓展性。

思科的 DistributedDirector[28]实现了一种自适应请求路由算法。该算法采用三种度量的加权组合，即 AS 间距、AS 内距和端到端延时。尽管使用三种度量后提高了灵活性，但计算复杂且代价高昂。另外，主动性的延时测量引入了额外的网络流量。更有甚者，DistributedDirector 的组件与副本服务器相分离，使得其不能探测服务器并得到服务器的负载信息。

Akamai[1,29]使用了一个复杂的请求路由算法，能够自适应地处理瞬时拥塞问题。该方法使用了多种度量，如副本服务器负载、客户端与每一台副本服务器负载的可靠性以及副本服务器的可用带宽等。该算法为 Akamai 专有，故算法实现细节并未公开。

2. 请求路由机制

请求路由机制使客户端清楚如何通过请求路由算法选择副本服务器。请求路由机制的分类可以按照若干标准来进行，这里依据的是对请求的处理，分为全局服务器负载均衡（GSLB）、基于 DNS 的请求路由、HTTP 重定向、URL 改写、任播（anycasting）、CDN 对等，如图 2.12 所示。

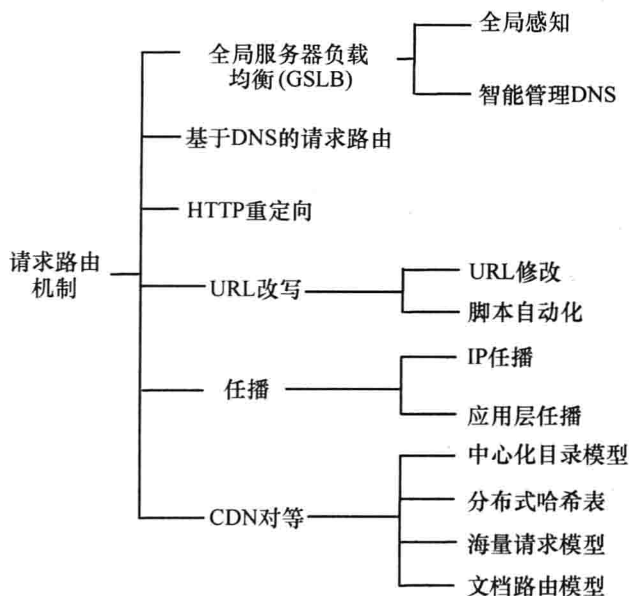


图 2.12 请求路由机制的分类

在 GSLB[44]方法中，提供服务的节点将内容发送到端用户，服务节点由一个具有 GSLB 功能的 Web 交换机和全球分布的大量 Web 服务器组成。该方法定义的服务节点具有两个新的能力，使其能支持全局服务器负载均衡。第一个能力是全局感知，第二个是智能管理 DNS[44]。在本地服务器的负载均衡处理中，每一个服务节点都可获知与其直连的 Web 服务器的状态和性能信息。在 GLSB 中，一个服务节点获知其他服务节点的信息，并将其虚拟 IP 地址存储在它的服务器列表中。因此，组成每一个服务节点的 Web 交换机是全局感知的，每一个服务节点都知道其他服务节点的地址，并在 GSLB 配置中通过交换机彼此交换性能信息。为了利用这些全局信息，GSLB 交换机充当某些特定域中的智能管理 DNS。GSLB 的优势在于，因为服务节点可以互相感知，每一个 GSLB Switch 均可为任何请求选择最优的代理缓存服务器。因此，该方法方便了服务器的选择，不仅仅可以从本地相连的实际服务器池中选择，还能从远程的服务节点中选择。GSLB 的另外一个优势在于，网络管理员能够在不需要额外添加网络设备的情况下增加网络的 GSLB 能力。GSLB 的不足是需要手动配置服务节点来使得它们能够与 GSLB 实现性能的匹配。

在基于DNS的请求路由方法中，内容分发服务通过修改DNS服务器来实现代理缓存服务器的符号名与其IP地址之间的映射。映射被用于“整体复制”模式下的内容选择和分发。在基于DNS的请求路由中，一个域名对应多个IP地址。当接收到一个端用户的内容请求时，服务提供商的DNS服务器返回拥有被请求对象副本的那些服务器的IP地址，客户端的DNS解析器从中选择一个服务器。为了判断哪一个服务器合适，解析器需要分配一个探测器给服务器，通过测量这些服务器到探测器的响应时间来确定具有的服务。它也可根据收集到的历史数据，如客户端对服务器的访问情况来选择服务器。采用“整体复制”和“部分复制”的CDN提供商都使用DNS重定向。基于DNS的请求路由系统的性能情况已在很多文献中进行了研究[15, 41, 65, 86]。这种方法的优势在于透明，因为对服务的调用是通过DNS域名而非IP地址。基于DNS的方法之所以得到广泛应用，原因在于简单易用，并且与任何实际的被复制的服务相独立，它被嵌入到域名解析服务中，所以可被任何互联网应用程序使用[89]。另外，提供目录服务的DNS到处可见，也方便了请求路由的实现。该方法的缺点在于增加了网络延时，因为它延长了DNS查找时间。CDN管理员一般通过将CDN的DNS分割为两层（低层DNS和高层DNS）来解决这个问题[58]。另外一个局限是DNS提供的是客户端当地DNS（LDNS）的IP地址，并不是客户端本身的IP地址。客户端被管理员假定位于当地DNS附近，当基于DNS的服务器选择方法被用于选择邻近服务器时，决策是由域名服务器而不是客户端来进行。因此，当客户端和域名服务器不在一起时，基于DNS的方法会产生性能低下的决策。更重要的是，DNS不能用于控制所有的访问请求，因为对DNS数据的缓存同时在ISP和客户端两个层面上进行。事实上，在很多实例中DNS仅能够控制至多5%的请求[20]。更有甚者，因为客户端并不连接到为其提供内容服务的实际域名，这使得一旦指定的代理缓存服务器失效，就没有任何其他服务器来为客户端的请求服务。因此，在网络或服务器状况不断变化的情况下，为了保证实时响应，基于DNS的方法必须避免在客户端进行缓存或决策。

HTTP重定向技术通过HTTP头来传播副本服务器集的信息。HTTP允许一个网络服务器通过一种特殊报文来响应客户端请求，该报文告知用户重新发送请求到另外的服务器。HTTP重定向可用于整体复制和部分复制的内容选择和分发。该机制可用于构建一种特殊的网络服务器，以接收用户的请求，并选择副本服务器和将用户重定向到这些服务器上。这种机制需要对网络服务器和客户端进行一些修改以处理额外的报文头。该方法的主要优点在于可拓展性和简易性，而且对复制过程可以进行精细粒度的管理，因为每一个网页可以被视为颗粒[75]。HTTP重定向技术最大的缺点在于缺乏透明度。更重要的是，它在HTTP和处理的请求上引入了额外的报文往返操作，因此系统开销非常大。

尽管大多数CDN系统使用基于DNS的路由机制，但也有一些系统使用URL的改写或者超链接，主要用于部分复制的内容选择和分发，其中内嵌对象应客户端的请求而发送。在此方法中，源服务器对动态生成的页面的URL链接进行改写，从而将客户端重定向到不同的代理缓存服务器。例如，一个网页包含HTML文件和一些内嵌对象，网页服务器会修改内嵌对象的索引列表，使客户端能够通过最优的代理缓存服务器获取这些内容。为了使该过程自动化，CDN使用特殊脚本来透明地解析网页的内

容和替换内嵌对象的 URL[58]。URL 的改写可以是先验式也可以是反应式的。在先验式的 URL 改写中，HTML 主页中内嵌对象的 URL 在内容被加载到源服务器之前生成。反应式方法在客户端请求到达源服务器后才进行 HTML 页面中内嵌 URL 的改写操作。URL 改写的主要优点是客户端不必受限于一台代理缓存服务器，因为改写的 URL 包含了指向一组代理缓存服务器的 DNS 名称。另外，通过这种方式也能达到更优的粒度，因为内嵌对象可以被认为是颗粒。该方法的缺点是 URL 解析带来的延时以及因为引入一个径内因素可能导致的瓶颈。另外一个劣势是那些访问索引被改为指向邻近代理缓存服务器（而非源服务器）的内容是不能缓存的。

任播法可以被分为 IP 任播和应用层任播。IP 任播由 Partridge 等人[73]提出，假定同一 IP 地址被指派到一组主机，每一个 IP 路由器保持路由表中一条从主机到路由器的最短路径。因此，不同 IP 路由器管理着通向不同主机（具有相同 IP 地址）的路径。IP 任播适合请求路由和服务定位，可以在潜在的异构平台上实现全网范围的服务器复制。IP 任播的缺点在于某些 IP 地址空间被分给了任播地址。Fei 等人[32]提出了一种应用层任播机制，其中服务包含了一组任播解析器，能进行任播域名到 IP 地址的映射。用户与任播解析器进行交互产生任播请求，解析器处理请求返回任播的响应结果。一个度量数据库与每一个任播解析器相连，包含副本服务器的性能数据。服务器的性能评估涉及服务器的负载和处理请求的能力，性能测量所带来的系统开销被保持在一个可管理水平上。性能数据可用于从一个服务器组中，根据用户特定性能要求来选择某一服务器。应用层任播的优势在于灵活性，而缺点是部署对于请求路由的任播机制时需要对服务器和客户端都进行修改，因此对于大量服务器和客户端而言增加了成本。

对等的内容网络通过主机之间的对称连接形成，形成对等 CDN 之间可以代为转发彼此的内容。因此，通过 CDN 服务器及其附近的代理服务器，一个 CDN 可以扩展自己的范围到更大的用户群中。一个内容提供商一般只与一个 CDN 有协议，每一个 CDN 再代表内容提供商与其他对等 CDN 互联[74]。对等 CDN 更加具有容错性，因为所有必要的信息检索网络都可以通过对等成员得到，而不必依靠像传统 CDN 那样的复杂结构。为了定位 CDN 对等成员的内容，可以使用一种中心化的目录模型或分布式哈希表（DHT），也可以使用洪泛请求模型或文档路由模型[44, 66]。

在中心化的目录模型中，所有对等节点将其持有的共享内容登记在中心目录中，并可以访问该目录。当目录收到请求时，它返回持有被请求内容的节点信息。当有不止一个节点可以服务请求时，最优的节点的选择可以根据网络距离、最大带宽、最小拥塞和最高容量等信息产生。收到目录发出的响应后，发出内容请求的节点从指定的最优节点获取内容。该方法的缺陷在于，中心化的目录易出现单点失效问题。另外，基于中心目录的系统，其可拓展性受目录容量的限制。Archi[31] 和 WAIS[52] 是中心化目录模型的典型例子，用于跨系统的 FTP 文件传输。

在使用 DHT 的系统中，使用哈希键值对分布式节点进行索引，因此持有被请求内容的节点可通过查询和查找函数得到[43]。使用 DHT 的例子是 Chord[92]，其优势在于一个过载的节点能够将自身的负载转移到负载较轻的节点，以便达到负载均衡[18]。在海量请求模型中，来自节点的请求被广播给与其直连的其他节点，这些节点再进一

步将报文转发给与其直连的节点，该过程一直持续到请求被响应或达到某种限制为止。该方法的缺陷在于它产生了额外的网络流量，因此对带宽的要求大大增加。因此，该方法受制于可拓展性和网络规模[44]。Gnutella[8,25]是一个使用海量请求模型系统的例子。在文档路由模型中，经过授权的节点得到被请求的内容。该模型中，每个节点负责提供一定范围的文件 ID，可以部分地提供被请求的内容[44]。当某个节点需要得到某个文件时，它便发送包含文件 ID 的请求。请求被转发给含有最接近文件 ID 号的节点，一旦文件被定位，该文件就被传输到发出请求的节点。该方法的主要优点是其算法复杂度为 $O(\log n)$，能在有限步内完成全局的搜索，并表现出良好的性能，且其优良的扩展性使网络规模可以非常庞大。

2.2.4 性能测量

CDN 能够测量其服务客户端请求的特定内容和（或）服务的能力。一般而言，内容提供商主要使用 5 种指标来评估 CDN 的性能[30,37,58]，包括：

（1）缓存命中率

该指标定义为被缓存文档的数量与被请求文档总数的比率。一个高的缓存命中率表明 CDN 使用了高效的缓存技术。

（2）保留带宽

该指标定义为源服务器使用的带宽，以字节为单位，通过检测源服务器获得。

（3）延时

该指标指用户感知的响应时间。延时的下降表明源服务器保留了较少的带宽。

（4）代理缓存服务器利用率

该指标指代理缓存服务器持续繁忙的时间占比，用于计算 CPU 负载、服务请求的数量和存储 I/O 的利用率。

（5）可靠性

CDN 的可靠性是根据丢包情况来判断的，高可靠性表明 CDN 丢包较少，并且对于客户端来说可用性较高。

性能测量可以基于系统内部的性能评定，也可以从用户的角度进行。一个 CDN 提供商本身的性能测量可能没有什么实际意义，因为对于某一特定网址该 CDN 性能可能表现良好，而对其他站点而言则可能一般。为了确保可靠的性能测量，一个 CDN 的性能可以通过独立的第三方来测试完成，如 Keynote 系统[3]或 Giga Information Group[6]。性能测量的分类如图 2.13 所示。

1. 内部测量

CDN 服务器可以收集统计数据，用于得到网站的端对端性能。除此之外，可以在整个网络中部署探测器（硬件和软件），通过分析探测器采集到的信息与服务器日志及缓存之间的相关性，就可以测量出端到端的性能。

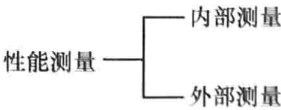


图 2.13 性能测量的分类

2. 外部测量

除内部性能测量外，通过独立的第三方性能测量可以从 CDN 用户的角度获得系统的实际性能。这个过程相当有效，因为独立的性能测试公司可以提供基准测量网络，该测量网络由分布在若干城市骨干网中按一定策略分布的测试计算机互联而成。这些计算机从终端用户的角度，使用一些关键的服务性能指标，来评定一个特定站点的性能表现[95]。

3. 用于性能测量的网络信息获取

对于内部和外部的性能测试，可以使用基于不同参数的网络统计信息获取技术，包括网络探测、流量监控和代理缓存服务器反馈。网络统计获取过程中典型的参数包括地域邻近度、网络邻近度、延时、服务器负载和服务器整体性能。图 2.14 显示了 CDN 使用的网络统计信息获取技术。

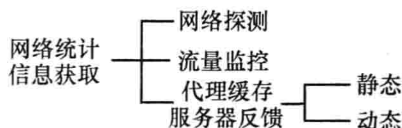


图 2.14 网络统计信息获取技术

网络探测是一种测量技术，用于检测那些可能发出请求的实体，以期从一个或多个代理缓存服务器处获取一种或多种测量指标。网络探测可用于基于 P2P 的协作 CDN 中，其中代理缓存服务器不由单一 CDN 提供商控制。ICMP ECHO 报文就是此类探测技术的一个例子，该报文被一个或多个代理缓存服务器周期性地发到潜在请求方。由于某些原因，主动探测技术在某些情况下并不适用，而且在有些方面具有局限性。该技术引入了额外的网络延时，可能对小型网络的影响较大。另外，针对一个实体执行多个探测器的操作经常引发冲突检测，导致用户的不满[35]。探测技术有时也会导致不准确的测量结果。因为担心遭到分布式拒绝服务攻击（DDoS），ICMP 流量有可能被忽略或被变更优先次序。Freedman 等[35]提出的分布式任播系统表明，如果使用高随机性的端口，ICMP 探测器和 TCP 探测器经常被防火墙拒绝或者被标记为非法的端口扫描。

流量监控是一种通过监控客户端和代理缓存服务器之间的流量来得知确切性能测量数据的检测方法。一旦客户端建立连接，传输过程的实际性能将被测量。测量数据反馈到请求路由系统，一个例子是通过测量 TCP 的性能来获取从客户端到代理缓存服务器的丢包情况或者用户感知延时的情况。延时是最简单也是最常用的距离测量指标，可通过客户端到代理缓存服务器之间路由路径的包传递（即流量）情况进行估计。测量估计系统（如 IDMaps[35]）在整个互联网范围内测量和传播延时和带宽等距离信息。该系统考虑两类及时性的距离信息，即负载敏感和“原始”信息（也就是说，距离信息是在不考虑网络中负载情况下获得的）。对这些信息进行估计时，以某种时间更新频率进行流量监控，更新频率可以按天计，如有必要也可以小时计。

代理缓存服务器的反馈信息可以通过周期性的探测得到，方式是发送与特定应用相关的请求（如 HTTP）以及采取相关的测量手段。反馈信息也可通过代理缓存服务器中部署的代理得到。这些代理可得到关于彼此节点的一系列测量信息。反馈信息的获得手段可以是静态的也可以是动态的。静态方法通过选择路由来最小化路径中的跳数或者通过优化其他静态参数得到。动态探测需要计算往返时间或实时的 QoS 参数[33]。

图 2.15 列出了 CDN 在测量网络和系统性能时使用的不同指标。地域邻近度是一

种判断特定区域内用户位置的技术，常用于将特定区域内的所有用户重定向到同一个接入点(POP)。这种网络邻近度的测量一般通过探测BGP路由表得到。端用户感知的延时也是一个有用的指标，可用于为用户选择合适的代理缓存服务器。一条网络路径的丢包信息也是一种指标，用于选择具有最小错误率的路径。平均带宽、启动时间和帧率等指标都可用于选择流媒体分发的最优路径。服务器负载状态可通过计算CPU负载、网络接口负载、活动连接和存储I/O负载等相关信息得到。服务器负载状态指标用于选择具有最小累积负载的服务器。



图2.15 用于测量网络和系统性能的指标

4. 通过仿真进行性能测量

除了内部和外部性能测量外，也可使用仿真工具来分析一个CDN的性能。一些研究人员使用了一些真实平台如PlanetLab[5]等进行他们自己设计的CDN策略实验。通过软件实现的CDN仿真器对于研究人员开发、测试和诊断CDN的性能来说非常有用。商业CDN的专有特性，使得访问真实CDN的信息和日志并不容易。在实验过程中无须引入复杂的硬件设备。因此仿真就显得经济实惠，另外，仿真也非常灵活，因为在仿真实验中可以模拟任何带宽和传输延时的连接，也可以模拟出任意队列大小的路由器和相关排队管理技术。一个虚拟化的网络环境避免了真实网络环境中研究人员不可避免的不可控因素，如额外的流量等。因此，仿真结果是可重现的，也容易进行分析。目前，已有多种网络仿真器[4,7]可用于模拟一个CDN的性能测量。此外，还有一些特殊的CDN仿真系统[2,7,23,54,100]，可供研究团体和CDN开发者以一种相当真实的方式来试验他们自己的策略和测量系统性能。然而，如果CDN系统不考虑诸多关键因素，如真实网络中经常遇见的问题（瓶颈问题或遍历节点数）TCP/IP网络架构，那么通过仿真得到的结果也可能具有误导性。

2.3 分类法到代表性CDN的映射

本节将提出的分类法对应到一些有代表性的CDN中，这些CDN已经在第1章进行了概述。在对现有系统进行分类的同时，本节还提供了对这些系统的相关见解和评估。通过分类法对CDN进行分析，反过来也验证了分类法的有效性和适用性。

2.3.1 基于CDN构成的分类

表2.1给出了基于CDN构成分类若干代表性CDN的资料。如表中所示，现有大多数CDN在组网时都使用覆盖法，也有使用网络法或两者兼具。无论覆盖法还是网络法，在Akamai和Mirror Image等商业CDN中都比较常见。当CDN提供商使用以上两者的组合来组建CDN时，可使用一个网络组件将HTTP请求重定向到邻近的实现特定应用的代理缓存服务器。

表2.1 CDN组件分类映射示意图

名称、类型		CDN组织	服务器	关系	交互协议	内容/服务类型
商业CDN	Akamai	网络法和覆盖法	源服务器和副本服务器	客户端—代理缓存服务器—源服务器 网络组件—缓存代理服务器，代理服务器	网络组件交互 缓存间交互	静态、动态内容流媒体和服务（网络监控，地域定位）
	Edge Stream	网络法	N/A	N/A	网络组件交互	视频流、视频托管服务
	Limelight Networks	覆盖法	源服务器和副本服务器	客户端—代理缓存服务器—源服务器 网络组件—缓存代理服务器	网络组件交互	静态内容、流媒体
	Mirror Image	网络法和覆盖法	源服务器和副本服务器	客户端—代理缓存服务器—源服务器 网络组件—缓存代理服务器	网络组件交互	静态内容、流媒体Web计算和报告服务
学术CDN	CoDeeN	覆盖法（带有“开放”代理服务器）	源服务器和副本/代理（前向、反向、重定向）服务器	客户端—代理缓存服务器—源服务器 网络组件—缓存代理服务器	网络组件交互 缓存间交互	对于大多数站点来说，参与用户能得到更高质量的服务；仅提供静态内容
	Coral	覆盖法（带有索引设施）	源和副本/（协作）代理缓存服务器	客户端—代理缓存服务器—源服务器 网络组件—缓存代理服务器	网络组件交互 缓存间交互	参与站点可以为大多数用户提供更高质量的服务；仅提供静态内容
	Globule	覆盖法（带有端用户节点）	源、副本、备份和/或重定向服务器	客户端—代理缓存服务器—源服务器 网络组件—缓存代理服务器	网络组件交互 缓存间交互	一个Web站点的性能和可用性得到提高；提供静态内容和监控服务

学术 CDN 通过使用覆盖法和 P2P 技术建立。然而，这些 CDN 在覆盖的建立和部署方式上各不相同。例如，CoDeeN 的覆盖使用了“开放式”的代理服务器，而 Coral 的覆盖（包括协作 HTTP 代理服务器和一个 DNS 服务器网络）依靠其下的索引设施建立，Globule 的覆盖由端用户节点组成。

在覆盖法中，常见的关系为客户端—代理缓存服务器—源服务器和网络组件—缓存代理服务器。代理服务器—代理服务器的关系在 CDN 中也很常见，代理服务器间关系支持缓存服务器间的交互。当使用网络法时，CDN 依赖于网络组件的交互，在通过预设策略放置的网络组件上部署请求路由逻辑来提供服务。覆盖法优于网络法的原因在于新的服务集成和简化的网络设施管理。对于覆盖法而言，提供一个新服务就像在 CDN 服务器中加入新代码一样简单[61]。

CDN 使用源服务器和副本服务器进行内容分发。大多数副本服务器都作为 Web 服务器来提供 Web 内容服务。某些 CDN 如 Akamai、EdgeStream、Limelight Networks 和 Mirror Image 等，将其副本服务器作为多媒体服务器来分发流媒体和视频托管服务。副本服务器也可以用于提供缓存、大文件传输、报告和 DNS 等服务。在学术 CDN 中，代理/副本服务器可以出于不同的目的进行配置。例如，每一个 CoDeeN 节点都能用做一个前向、反向或重定向代理服务器。Coral 代理服务器之间彼此协作，而 Globule 节点可以作为源服务器、副本服务器或备份服务器使用。

从表 2.1 中可以看出，大多数 CDN 网络的目标还是用于提供特定内容。因为服务和内容的差异，需要 CDN 具备与特定应用相关的特点、结构和技术。大多数 CDN 仅提供静态内容，少数 CDN 提供流媒体、广播和其他服务。对于商业 CDN 而言，其目的是通过内容和（或）服务分发来盈利。然而学术 CDN 的目的却各不相同。例如，CoDeeN 提供静态内容的目的是向其用户访问大多数网站时提供更好的性能；Coral 的目的在于提高用户在访问大多数 Web 站点时的性能体验；Globule 力图提升用户访问 Web 站点的性能体验和可用性，以及在某种程度上应对瞬时拥塞和 Slashdot 效应的能力。

2.3.2 基于内容分发和管理的分类

表 2.2 中给出了内容分发和管理分类法与代表性的 CDN 之间的映射关系。大多数 CDN 支持部分复制方式下的内容分发，有的 CDN 也同时支持整体复制和部分复制。CDN 提供商倾向于使用部分复制的内容分发，因为这样做能够降低源服务器的负载以及相应的设施成本。另外，因为内嵌型内容的非频繁变化性，部分复制的方法要比整体复制的性能更好。仅仅一小部分 CDN（如 Akamai、Mirror Image 和 Coral 等）支持内容的聚类技术，其他 CDN 的内容分发设施架构并没有显示其是否使用了内容聚类技术。Akamai 和 Coral 通过用户会话期进行内容聚类，此方法的有效之处是有助于辨识相似浏览模式的用户群和含有相关内容信息的那些页面。唯一使用 URL 进行内容聚类的是 Mirror Image，但是基于 URL 的方法并非主流，原因在于该方法受限于其部署的复杂性。

通过表 2.2 能够清晰地看出，大多数覆盖大范围地域的代表性 CDN 都使用多 ISP 法，以在众多全球性 ISP 的接入点处放置大量代理缓存服务器。商业 CDN（如 Akamai、Limelight Networks 和 Mirror Image）和学术 CDN（如 Coral[34] 和 CoDeeN）均使用

多 ISP 法。因为单 ISP 法具有这样的缺陷，即代理缓存服务器相距较远而终端用户之间则相对较近。然而，多 ISP 法中部署和管理系统以及建网的费用、管理成本和复杂度都要更高。当然，对于那些具有高流量的站点，以上问题可以不用考虑。相对于单 ISP 法，多 ISP 法表现更佳的原因是单 ISP 法只适用于低到中等流量的站点。

表 2.2　内容分发和管理分类映射示意图

名称	内容选择和分发	代理放置	内容外包	缓存组织
Akamai	内容选择 • 全部和部分站点内容分发 内容聚类 • 基于用户会话期	多 ISP 方法；热点放置（通过对高负载站点放置更多的服务器）	非协作拉取法	缓存技术 • 类内、类间缓存 缓存更新 • 更新传播 • 按需更新
Edge Stream	内容选择 • 部分站点内容分发 内容聚类 N/A	单 ISP 方法	非协作拉取法	缓存技术 • 类间缓存 缓存更新 N/A
Limelight Networks	内容选择 • 部分站点内容分发 内容聚类 N/A	多 ISP 方法	非协作拉取法	缓存技术 • 类内缓存 缓存更新 • 按需更新
Mirror Image	内容选择 • 部分站点内容分发 内容聚类 • 基于 URL	多 ISP 方法； 通过集中化的“超市”架构实现中心放置	非协作拉取法	缓存技术 • 类内缓存 缓存更新 • 按需更新
CoDeeN	内容选择 • 部分站点内容分发 内容聚类 N/A	多 ISP 方法； 拓扑已知的副本放置	协作拉取法	缓存技术 • 类内、类间缓存 缓存更新 • 按需更新
Coral	内容选择 • 全部和部分站点内容分发 内容聚类 • 基于用户会话期	多 ISP 方法；基于树的副本放置	协作拉取法	缓存技术 • 类内、类间缓存 缓存更新 • 缓存失效
Globule	内容选择 • 部分站点内容分发 内容聚类 N/A	单 ISP 方法；最好的副本置放策略是通过对于不同的策略的定期评估来动态选择	协作拉取法	缓存技术 • 类内、类间缓存 缓存更新 • 自适应性缓存更新

商业 CDN 的内容外包机制多使用非协作拉取法，原因在于 DNS 重定向和 URL 改写使得该方法简单易行。协作推送法仍然处于理论阶段，目前还没有 CDN 支持这种方法。相对于非协作推送法，协作推送法需要使用更复杂的技术（如 DHT），一般用于 P2P 架构的学术 CDN 中[71]。另外，当用户数量很大时，此方法具有较大的通信开销（彼此交换报文）。并且，当内容更新较为频繁或一致性要求比较苛刻时，该方法也并不能提供高可靠性的保证。

从表 2.2 中也能发现，具有代表性的覆盖大地域的商业和学术 CDN 使用类间（以及类内和类间的组合）缓存技术。CDN 主要将按需更新作为缓存更新策略。只有 Coral 将缓存的无效性作为缓存更新策略，因为它只传递几乎不更新的静态内容。Globule 使用一种自适应性的缓存更新策略，可以在增强缓存一致性的不同技术之间实现动态的选择。在所有缓存更新策略中，周期性更新方法应用最为广泛，因为在此情况下缓存以有规律的方式进行更新，因此该方法具有成为最有效地确保缓存内容一致性的潜力策略。更新的传播和失效作为稳态控制机制并不常用，因为它们挤占用了本应用于内容服务的网络带宽和处理器资源[41]。内容提供商可能会自行管理和部署特定的缓存机制或缓存更新的启发性策略。特定的分布式缓存机制更易于管理，但是效果有限。另外，对于不愿自主开发缓存机制的内容提供商而言，启发式缓存策略是一个非常好的策略。然而，启发式方法取得的效果无法与那些设计良好的策略控制相比[41]。

2.3.3 基于请求路由的分类

表 2.3 将请求路由分类法映射到具有代表性的 CDN 中。从表中可以看到，基于 DNS 的机制是实现请求路由的主流方法，原因在于实现简单，而且作为目录服务的 DNS 无处不在。基于 DNS 的机制在域名解析过程中使用了专用的 DNS 服务器。在其他请求路由机制中，也常用 HTTP 重定向，因为用户和副本服务器间的显式绑定成本具有更细的粒度。灵活性和易用性是 HTTP 重定向技术用于 CDN 请求路由的另外两种原因。有些 CDN（如 Mirror Image）使用 GSLB 来进行请求路由，该方法的优点是将 GSLB 功能引入网络时，不需要添加任何网络设备。在学术 CDN 中，Coral 探索了覆盖路由技术，其中使用 DSHT 为请求路由的摘要进行索引，也就是说，它在请求重定向中使用了 P2P 技术。如之前提到的那样，一个 CDN 的请求路由系统是由一个请求路由算法和请求路由机制组成的。CDN 中的请求路由算法是专有的，绝大多数的技术细节没有公开。本章对现有 CDN 的分析表明，Akamai 和 Globule 在请求路由系统中使用了自适应请求路由算法。Akamai 的自适应性请求路由机制（应对瞬时拥塞）考虑到了服务器负载和多种网络指标，而 Globule 仅仅测量了一个请求需要经过的 AS 数量。对于 CoDeeN 而言，请求路由算法考虑到了请求的本地性、系统负载、可靠程度和邻近信息。另外，Coral 的请求路由算法通过测量实时网络性能数据来提升本地性，并存储拓扑信息用于增加客户端发现邻近 DNS 服务器的可能性。

表 2.3　请求路由分类法映射示意图

CDN 名称	请求路由技术
Akamai	• 自适应请求路由算法（使用服务器负载和各种网络指标） • 综合了基于 DNS 的请求路由和 URL 改写
EdgeStream	HTTP 重定向
Limelight Networks	基于 DNS 的请求路由
Mirror Image	全局服务器负载均衡（GSLB） • 全局信息已知 • 智能管理 DNS
CoDeeN	• 请求路由算法使用请求的本地性、系统负载、可靠程度和邻近信息 • HTTP 重定向
Coral	• 局部性得到改进的请求路由算法（通过测量网络的最新性能数据和存储网络拓扑信息） • 基于 DNS 的请求路由
Globule	• 自适应请求路由算法（使用基于 AS 的邻近度） • 单层基于 DNS 的请求路由

2.3.4　基于性能测量的分类

表 2.4 显示了不同性能测试技术与代表性 CDN 的映射关系。

通过某些参数估计技术测量 CDN 的性能能够衡量 CDN 为其客户提供所需内容和（或）服务的能力。衡量 CDN 性能的指标应当包括缓存命中率、带宽消耗、延时和代理缓存服务器的利用率等。此外，其他因素如存储、通信系统开销和可扩展性也应当考虑在内。对于性能度量的估计能够表明系统的当前状况，并有助于在大系统中获得高效的请求路由和负载均衡。对于内容提供商而言，应该通过对一个 CDN 的性能研究来选择最合适的 CDN 提供商。然而，CDN 提供商的和有封闭特性，不允许内容提供商对网络进行性能测试。

从表 2.4 中可以发现，CDN 的性能测试可以通过内部测量技术和来自外部的用户评价得到。大多数 CDN 都使用基于网络探测、流量监控等相关技术进行内部性能测试。Akamai 使用预先的流量监控和网络探测进行性能测量。在学术 CDN 领域，CoDeeN 具备本地监控能力，可检测一个服务的主要资源，如可用的文件描述符/套接字、CPU 负载和 DNS 解析服务等；Coral 能够在响应一个 DNS 请求之前检测代理服务器是否可用（通过 UDP 远程过程调用，即 RPC）；Globule 在用于检查其他服务器可用程度的重定向服务器上实现监控。

CDN 提供商性能的外部测试并不常见，因为大多数运营中的 CDN 都是非透明运营的商业企业，这些企业不公开其性能测量的相关方法是处于利益的考虑。然而，某些 CDN 如 Akamai 也允许第三方来进行外部测试。

表 2.4 性能测量分类映射示意

CDN 名字	性能测量
Akamai	内部测量 • 网络探测 • 流量监控（周期性） 外部测量 • 通过第三方进行检测（Giga information group）
Edge Stream	内部测量 • 通过实时性能监控服务（RPMS）进行流量监控
Limelight Networks	N/A
Mirror Image	内部测量 • 网络探测 • 流量监控和报告
CoDeeN	内部测量 • 本地流量和系统监控
Coral	内部测量 • 流量监测 • 通过 UDP RPC 对于服务器进行生存性检测
Globule	内部测量 • 流量监测 • 通过重定向器来检测服务器可用性

2.4 讨论

如本章开始所述，除了分类法中主要论及的四个核心内容，一个完备的 CDN 需要考虑诸多其他因素，如容错、安全和 Web 应用托管能力等。本章基于以上话题进行了简要探讨，并提供了相关的研究资料，以期读者能够对相关研究领域的内容有一个概括性的把握。

CDN 作为一个复杂的由分布式组件构成的网络，故障和失效的现象可能会发生在任何地方，使用局部聚集等中心化架构有助于提高容错能力。然而，当 ISP 到服务器集群的连接中断时，单点失效就会发生。这个问题可以通过部署分布式的 Web 集群（镜像）或者使用多 ISP 来提高连接的可靠性（多路寻址或多路连接）。然而，尽管集群、镜像或多路寻址等方法在某种程度上提高了 CDN 的鲁棒性，但也同时引入了其他的问题。如，集群的可扩展性不佳，镜像技术需要每一个镜像都承担所有的负载，多路寻址技术需要每一个连接都承担整体流量。因此，不同的商业 CDN 使用其独特的方法来提高容错性和可扩展性。例如，Akamai 开发了分布式监控服务，确保服务器或者网络故障能在第一时间得到处理，从而不对端用户造成影响。除此之外，参考文献中还提出了

大量相关解决方案，已在一些系统中得到了广泛应用。感兴趣的读者可在参考文献[46, 77, 85]中找到大规模系统中（如 CDN 中）的相关容错方法。

CDN 中的安全问题对系统开发来说是另一个挑战。在 CDN 的不同层面上（如网络、路由、DNS 或 Web 集群等）会有不同的安全隐患，一个常见的威胁就是 DDoS 攻击。DDoS 攻击不仅会消耗稀缺资源如网络带宽或 CPU 等，而且也会破坏或修改信息配置，或对网络组件造成物理性损坏或者篡改[82]。安全威胁还包括针对软件脆弱性的攻击（入侵攻击）和针对协议不一致性的攻击（协议攻击）。目前，已经有很多工作涉及这些大范围网络存在的安全问题，相关解决方案的资料可参见参考文献[40, 50, 51, 53, 56, 57, 82]。

如今，Akamai 等商业 CDN 已经给端用户提供了基于使用的内容和应用的分发方案。Akamai 边缘计算架构（ECI）[1]、主动缓存[19]和 ACDN[80]等将数据保存在中心服务器中，把应用代码而非应用数据复制到边缘服务器，这可以将 Web 层的那些应用拓展到 CDN 平台上，使端用户对应用对象的请求可以直接在副本服务器而不是在源服务器上执行。然而，大量的数据访问造成了持续增长的大范围延时，另外，中心服务器数据的高度集中也会导致瓶颈。为了解决这些局限性，Sivasubramanian 等人[87]提出了一种按需复制 Web 应用的方法，该方法将数据进行部分复制，即只将数据复制到经常访问该数据的服务器上。参考文献[39]中提出了一种针对特定应用的边缘服务架构，该应用可以在一个弱一致性模型约束下，由其自己负责复制工作。更多关于在大范围网络中托管应用的研究可参阅参考文献[88]。

2.5 总结和结论

基于功能性和非功能性的属性，本章对 CDN 进行了分析和分类，提出了一种基于四个核心内容的 CDN 综合分类机制：CDN 构成、内容分发和管理、请求路由及性能测试。针对这些概念，本章进一步提出了相应的分类法，用于区分 CDN 中的若干主流趋势、解决方案和技术。另外，本章研究了容错、安全和 Web 应用托管能力 3 个问题，以此阐述在 CDN 开发中遇到的若干挑战，并提出了相关领域内的研究方向。本章提出的分类法为现有 CDN 之间的比较提供了基础，所提及的方法有助于读者深入了解领域内主流的技术、服务、策略和具体实现。本章也将分类法与代表性的商业 CDN 和学术 CDN 系统进行了对应，该对应关系有助于深入了解内容分发领域内的技术现状，并反过来验证了分类法的可行性和准确性。

最近，通过企业兼并，CDN 业界正在经历一场整合。在本章的前期准备工作中，作者深刻体会到这种整合给 CDN 行业带来的深刻影响。为此，为了应对下一代 CDN 的新技术和新架构的需要，内容分发、缓存和复制技术正在得到越来越多的关注，这将导致 CDN 的设计、架构和开发中产生新问题。内容网络领域的现有趋势表明，更好地理解和诠释该领域内的相关基础概念是非常必要的。因此，作者希望这种基于分类法的全面综合性比较框架，不仅可成为理解内容分发网这个复杂领域的工具，更有助于展现内容分发网未来研究的成果。

致谢

作者感谢本章中提到的商业和学术 CDN 的所有开发者，同时也要感谢那些未留下姓名的审阅者，他们中肯的评论和建议为本章增色匪浅。另外，作者要感谢墨尔本大学的同事 James Broberg、Marcos Assunção 和 Charity Lourdes，感谢他们给予本章的启示、评价和建议。感谢希腊塞萨洛尼基亚里士多德大学的 Athena Vakali、塞浦路斯大学的 George Pallis、意大利 ICAR－CNR 的 Carlo Mastroianni、意大利卡拉布里亚大学的 Giancarlo Fortino、意大利热那亚大学的 Christian Vecchiola 和美国普林斯顿大学的 Vivek Pai，他们在本章的分类问题方面提出了宝贵意见。作者还要感谢 Akamai 公司的 Fahim Husain、Mirror Image Internet 公司的 William Good 和 IBM T. J. Watson 研究中心的 Lisa Amini 在本文撰稿期间提供的有价值的研究论文、技术报告、白皮书和数据资料。

参考文献

[1] Akamai Technologies, 2007. www.akamai.com
[2] CDNSim, A Content Distribution Network Simulator, 2007. http://oswinds.csd.auth.gr/~cdnsim/
[3] Keynote Systems—Web and Mobile Service Performance Testing Corporation, 2007. http://www.keynote.com/
[4] Network simulators, 2007. http://www-nrg.ee.lbl.gov/kfall/netsims.html
[5] PlanetLab Consortium, 2007. http://www.planet-lab.org/
[6] The GigaWeb Corporation, 2007. http://www.gigaWeb.com/
[7] The network simulator – ns-2, 2007. http://www.isi.edu/nsnam/ns/
[8] Aberer, K. and Hauswirth, M. An overview on peer-to-peer information systems. In *Proc. of the Workshop on Distributed Data and Structures (WDAS)*, France, 2002.
[9] Aggarwal, A. and Rabinovich, M. Performance of dynamic replication schemes for an Internet hosting service. Technical Report, HA6177000-981030-01-TM, AT&T Research Labs, Florham Park, NJ, USA, 1998.
[10] Andrews, M., Shepherd, B., Srinivasan, A., Winkler, P., and Zane, F. Clustering and server selection using passive monitoring. In *Proc. of IEEE INFOCOM*, NY, USA, 2002.
[11] Androutsellis-Theotokis, S. and Spinellis, D. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4), ACM Press, NY, USA, pp. 335–371, 2004.
[12] Ardaiz, O., Freitag, F., and Navarro, L. Improving the service time of Web clients using server redirection. *ACM SIGMETRICS Performance Evaluation Review*, 29(2), ACM Press, NY, USA, pp. 39–44, 2001.
[13] Bakiras, S. and Loukopoulos, T. Combining replica placement and caching techniques in content distribution networks. *Computer Communications*, 28(9), pp. 1062–1073, 2005.
[14] Balakrishnan, H., Kaashoek, M. F., Karger, D., Morris, R., and Stoica, I. Looking up data in P2P systems. *Communications of the ACM*, 46(2), ACM Press, NY, USA, pp. 43–48, 2003.
[15] Barbir, A., Cain, B., Nair, R., and Spatscheck, O. Known content network request-routing mechanisms. Internet Engineering Task Force RFC 3568, 2003. www.ietf.org/rfc/rfc3568.txt
[16] Bartal, Y. Probabilistic approximation of metric space and its algorithmic applications. In *Proc. of 37th Annual IEEE Symposium on Foundations of Computer Science*, 1996.
[17] Brussee, R., Eertink, H., Huijsen, W., Hulsebosch, B., Rougoor, M., Teeuw, W., Wibbels, M., and Zandbelt, H. Content distribution network state of the art," Telematica Instituut, 2001.
[18] Byers, J., Considine, J., and Mitzenmacher, J. Simple load balancing for distributed hash tables. In *Proc. of 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, pp. 31–35, 2003.

[19] Cao, P., Zhang, J., and Beach, K. Active cache: Caching dynamic contents on the Web. In *Proc. of the Middleware Conference*, pp. 373–388, 1998.
[20] Cardellini, V., Casalicchio, E., Colajanni, M., and Yu, P. S. The state of the art in locally distributed Web-server systems. *ACM Computing Surveys*, 34(2), ACM Press, NY, USA, pp. 263–311, 2002.
[21] Chen, Y., Katz, R. H., and Kubiatowicz, J. D. Dynamic replica placement for scalable content delivery. In *Proc. of International Workshop on Peer-to-Peer Systems (IPTPS 02)*, LNCS 2429, Springer-Verlag, pp. 306–318, 2002.
[22] Chen, C. M., Ling, Y., Pang, M., Chen, W., Cai, S., Suwa, Y., Altintas, O. Scalable request-routing with next-neighbor load sharing in multi-server environments. In *Proc. of the 19th International Conference on Advanced Information Networking and Applications*, IEEE Computer Society, Washington, DC, USA, pp. 441–446, 2005.
[23] Chen, Y., Qiu, L., Chen, W., Nguyen, L., and Katz, R. H. Efficient and adaptive Web replication using content clustering. *IEEE Journal on Selected Areas in Communications*, 21(6), pp. 979–994, 2003.
[24] Cieslak, M., Foster, D., Tiwana, G., and Wilson, R. Web cache coordination protocol version 2. http://www.Web-cache.com/Writings/Internet-Drafts/draft-wilson-wrec-wccp-v2-00.txt
[25] Clip2 Distributed Search Solutions, The Gnutella Protocol Specification v0.4. www.content-networking.com/papers/gnutella-protocol-04.pdf
[26] Cooper, I., Melve, I., and Tomlinson, G. Internet Web replication and caching taxonomy. Internet Engineering Task Force RFC 3040, 2001. www.ietf.org/rfc/rfc3040.txt
[27] Davison, B. D. Web caching and content delivery resources. http://www.Web-caching.com, 2007.
[28] Delgadillo, K. Cisco DistributedDirector, Cisco Systems, Inc., 1997.
[29] Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. Globally distributed content delivery. *IEEE Internet Computing*, pp. 50–58, 2002.
[30] Douglis, F. and Kaashoek, M. F. Scalable Internet services. *IEEE Internet Computing*, 5(4), pp. 36–37, 2001.
[31] Emtage, A. and Deutsch, P. Archie: an electronic directory service for the Internet. In *Proc. of the Winter Usenix Conference*, San Francisco, CA, USA, pp. 93–110, January 1992.
[32] Fei, Z., Bhattacharjee, S., Zugura, E. W., and Ammar, M. H. A novel server selection technique for improving the response time of a replicated service. In *Proc. of IEEE INFOCOM*, San Francisco, California, USA, pp. 783–791, 1998.
[33] Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., and Zhang, L. IDMaps: a global Internet host distance estimation service. *IEEE/ACM Transactions on Networking (TON)*, 9(5), ACM Press, NY, USA, pp. 525–540, 2001.
[34] Freedman, M. J., Freudenthal, E., and Mazières, D. Democratizing content publication with Coral. In *Proc. of 1st USENIX/ACM Symposium on Networked Systems Design and Implementation*, San Francisco, CA, USA, 2004.
[35] Freedman, M. J., Lakshminarayanan, K., and Mazières, K. OASIS: anycast for any service. In *Proc. of 3rd Symposium of Networked Systems Design and Implementation (NSDI'06)*, Boston, MA, USA, 2006.
[36] Fujita, N., Ishikawa, Y., Iwata, A., and Izmailov, R. Coarse-grain replica management strategies for dynamic replication of Web contents. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 45(1), pp. 19–34, 2004.
[37] Gadde, S., Chase, J., and Rabinovich, M. Web caching and content distribution: a view from the interior. *Computer Communications*, 24(2), pp. 222–231, 2001.
[38] Gadde, S., Rabinovich, M., and Chase, J. Reduce, reuse, recycle: an approach to building large Internet caches. In *Proc. of 6th Workshop on Hot Topics in Operating Systems*, pp. 93–98, 1997.
[39] Gao, L., Dahlin, M., Nayate, A., Zheng, J., and Iyengar, A. Application specific data replication for edge services. In *Proc. of the Twelfth International World-Wide Web Conference*, Hungary, pp. 449–460, 2003.
[40] Garg, A. and Reddy, A. L. N. Mitigating denial of service attacks using qos regulation. In *Proc. of International Workshop on Quality of Service (IWQoS)*, 2002.
[41] Gayek, P., Nesbitt, R., Pearthree, H., Shaikh, A., and Snitzer, B. A Web content serving utility. *IBM Systems Journal*, 43(1), pp. 43–63, 2004.

[42] Hamilton, M., Rousskov, A., and Wessels, D. Cache digest specification – version 5. 1998. http://www.squid-cache.org/CacheDigest/cache-digest-v5.txt

[43] Harren, M., Hellerstein, J. M., Huebsch, R., Loo, B. T., Shenker, S., and Stoica, I. Complex queries in DHT-based peer-to-peer networks. In *Proc. of 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*, 2002.

[44] Hofmann, M. and Beaumont, L. R. *Content Networking: Architecture, Protocols, and Practice*. Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 129–134, 2005.

[45] Huffaker, B., Fomenkov, M., Plummer, D. J., Moore, D., and Claffy, K. Distance metrics in the Internet. In *Proc. of IEEE International Telecommunications Symposium*, IEEE CS Press, Los Alamitos, CA, USA, 2002.

[46] Jalote, P. *Fault Tolerance in Distributed Systems*. Prentice Hall, Englewood Cliffs, NJ, USA, 1994.

[47] Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., and Zhang, L. On the placement of Internet instrumentation. In *Proc. of IEEE INFOCOM*, Tel-Aviv, Israel, pp. 295–304, 2000.

[48] Jamin, S., Jin, C., Kure, A. R., Raz, D., and Shavitt, Y. Constrained mirror placement on the Internet. In *Proc. of IEEE INFOCOM*, Anchorage, Alaska, USA, 2001.

[49] Johnson, K. L., Carr, J. F., Day, M. S., and Kaashoek, M. F. The measured performance of content distribution networks. *Computer Communications*, 24(2), pp. 202–206, 2001.

[50] Jung, J., Krishnamurthy, B. and Rabinovich, M. Flash crowds and denial of service attacks: Characterization and implications for CDNs and Web sites. In *Proc. of the International World Wide Web Conference*, Hawaii, USA, pp. 252–262, 2002.

[51] Jung, J., Paxson, V., Berger, A. W., and Balakrishnan, H. Fast portscan detection using sequential hypothesis testing. In *Proc. of IEEE Symposium on Security and Privacy*, Oakland, 2004.

[52] Kahle, B. and Medlar, A. An information system for corporate users: wide area information servers. ConneXions—The Interoperability Report, 5(11), November 1991.

[53] Kandula, S., Katabi, D., Jacob, M., and Berger, A. W. Botz-4-sale: Surviving organized ddos attacks that mimic flash crowds. In *Proc. of Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, 2005.

[54] Kangasharju, J., Roberts, J., and Ross, K. W. Object replication strategies in content distribution networks. *Computer Communications*, 25(4), pp. 367–383, 2002.

[55] Karger, D., Sherman, A., Berkheimer, A., Bogstad, B., Dhanidina, R., Iwamoto, K., Kim, B., Matkins, L., and Yerushalmi, Y. Web caching with consistent hashing. *Computer Networks*, 31(11–16), pp. 1203–1213, 1999.

[56] Kargl, F., Maier, J., and Weber, M. Protecting Web servers from distributed denial of service attacks, In *Proc. of the International World Wide Web Conference*, pages 514–524, Hong Kong, 2001.

[57] Kim, Y., Lau, W. C., Chuah, M. C., and Chao, H. J. Packetscore: Statistics based overload control against distributed denial-of-service attacks. In *Proc. of INFOCOM*, Hong Kong, 2004.

[58] Krishnamurthy, B., Willis, C., and Zhang, Y. On the use and performance of content distribution network. In *Proc. of 1st International Internet Measurement Workshop*, ACM Press, pp. 169–182, 2001.

[59] Krishnan, P., Raz, D., and Shavitt, Y. The cache location problem. *IEEE/ACM Transaction on Networking*, 8(5), 2000.

[60] Kung, H. T. and Wu, C. H. Content networks: taxonomy and new approaches. *The Internet as a Large-Scale Complex System*, (Kihong Park and Walter Willinger eds.), Oxford University Press, 2002.

[61] Lazar, I. and Terrill, W. Exploring content delivery networking. *IT Professional*, 3(4), pp. 47–49, 2001.

[62] Lee, J. An End-User Perspective on File-Sharing Systems. *Communications of the ACM*, 46(2), ACM Press, NY, USA, pp. 49–53, 2003.

[63] Li, B., Golin, M. J., Italiano, G. F., Xin, D., and Sohraby, K. On the optimal placement of Web proxies in the Internet. In *Proc. of IEEE INFOCOM*, NY, USA, pp. 1282–1290, 1999.

[64] Ma, W. Y., Shen, B., and Brassil, J. T. Content services network: architecture and protocols. In *Proc. of 6th International Workshop on Web Caching and Content Distribution (IWCW6)*, 2001.

[65] Mao, Z. M., Cranor, C. D., Douglis, F., Rabinovich, M., Spatscheck, O., and Wang, J. A precise and efficient evaluation of the proximity between Web clients and their Local DNS servers. In *Proc. of the USENIX 2002 Annual Technical Conference*, Monterey, CA, USA, pp. 229–242, 2002.

[66] Milojicic, D. S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., and Xu, Z. Peer-to-peer computing. Technical Report, HP Laboratories, Palo Alto, CA, HPL-2002-57, 2002. www.hpl.hp.com/techreports/2002/HPL-2002-57.pdf

[67] Ni, J. and Tsang, D. H. K. Large scale cooperative caching and application-level multicast in multimedia content delivery networks. *IEEE Communications*, 43(5), pp. 98–105, 2005.

[68] Ni, J., Tsang, D. H. K., Yeung, I. S. H., and Hei, X. Hierarchical content routing in large-scale multimedia content delivery network. In *Proc. of IEEE International Conference on Communications (ICC)*, pp. 854–859, 2003.

[69] Pai, V. S., Aron, M., Banga, G., Svendsen, M., Druschel, P., Zwaenepoel, W., Nahum, E. Locality-aware request distribution in cluster-based network servers. *ACM SIGPLAN Notices*, 33(11), ACM Press, NY, USA, pp. 205–216, 1998.

[70] Pallis, G., Stamos, K., Vakali, A., Sidiropoulos, A., Katsaros, D., and Manolopoulos, Y. Replication-based on objects load under a content distribution network. In *Proc. of the 2nd International Workshop on Challenges in Web Information Retrieval and Integration (WIRI)*, Altanta, Georgia, USA, 2006.

[71] Pallis, G. and Vakali, A. Insight and perspectives for content delivery networks. *Communications of the ACM*, 49(1), ACM Press, NY, USA, pp. 101–106, 2006.

[72] Pallis, G., Vakali, A., Stamos, K., Sidiropoulos, A., Katsaros, D., and Manolopoulos, Y. A latency-based object placement approach in content distribution networks. In *Proc. of the 3rd Latin American Web Congress (La-Web 2005)*, IEEE Press, Buenos Aires, Argentina, pp. 140–147, 2005.

[73] Partridge, C., Mendez, T., and Milliken, W. Host anycasting service. Internet Engineering Task Force RFC 1546, 1993. www.ietf.org/rfc/rfc1546.txt

[74] Pathan, M., Broberg, J., Bubendorfer, K., Kim, K. H., and Buyya, R. An Architecture for Virtual Organization (VO)-Based Effective Peering of Content Delivery Networks, UPGRADE-CN'07. In *Proc. of the 16th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, Monterey, California, USA, 2007.

[75] Peng, G. CDN: Content distribution network. Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, Stony Brook, NY, 2003. http://citeseer.ist.psu.edu/peng03cdn.html

[76] Pierre, G. and van Steen, M. Globule: a collaborative content delivery network. *IEEE Communications*, 44(8), 2006.

[77] Pradhan, D. *Fault-Tolerant Computer System Design*. Prentice Hall, Englewood Cliffs, NJ, USA, 1996.

[78] Qiu, L., Padmanabhan, V. N., and Voelker, G. M. On the placement of Web server replicas. In *Proc. of IEEE INFOCOM*, Anchorage, Alaska, USA, pp. 1587–1596, 2001.

[79] Rabinovich, M. and Spatscheck, O. *Web Caching and Replication*, Addison Wesley, USA, 2002.

[80] Rabinovich, M., Xiao, Z., and Agarwal, A. Computing on the edge: A platform for replicating internet applications. In *Proc. of the Eighth International Workshop on Web Content Caching and Distribution*, Hawthorne, NY, USA, 2003.

[81] Radoslavov, P., Govindan, R., and Estrin, D. Topology-informed Internet replica placement. In *Proc. of Sixth International Workshop on Web Caching and Content Distribution*, Boston, Massachusetts, 2001.

[82] Ranjan, S., Swaminathan, R., Uysal, M., and Knightly, E. DDoS-Resilient scheduling to counter application layer attacks under Imperfect Detection. In *Proc. of INFOCOM*, pp. 1–13, 2006.

[83] Rousskov, A. and Wessels, D. Cache digests. *Computer Networks and ISDN Systems*, 30(22), pp. 2155–2168, November 1998.

[84] Saroiu, S., Gummadi, K. P., Dunn, R. J., Gribble, S. D., and Levy, H. M. An analysis of Internet content delivery systems. *ACM SIGOPS Operating Systems Review*, 36, pp. 315–328, 2002.

[85] Schneider, F. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial, 1 *ACM Computing Surveys*, 22(4), pp.299–320, 1990.

[86] Shaikh, A., Tewari, R., and Agrawal, M. On the effectiveness of DNS-based server selection." In *Proceedings of IEEE INFOCOM*, Anchorage, AK, USA, pp. 1801–1810, April 2001.

[87] Sivasubramanian, S., Pierre, G., and van Steen, M. Replicating Web applications on-demand. In *Proc. of IEEE International Conference on Services Computing (SCC)*, pp. 227–236, China, 2004.

[88] Sivasubramanian, S., Pierre, G., van Steen, M., and Alonso, G. Analysis of caching and replication strategies for Web applications. *IEEE Internet Computing*, 11(1), pp. 60–66, 2007.

[89] Sivasubramanian, S., Szymaniak, M., Pierre, G., and Van Steen, M. Replication of Web hosting systems. *ACM Computing Surveys*, 36(3), ACM Press, NY, USA, 2004.

[90] Stamos, K., Pallis, G., Thomos, C., and Vakali, A. A similarity-based approach for integrated Web caching and content replication in CDNs. In *Proc. of 10th International Databased Engineering and Applications Symposium (IDEAS 2006)*, IEEE Press, New Delhi, India, 2006.

[91] Stamos, K., Pallis, G., and Vakali, A. Integrating caching techniques on a content distribution network. In *Proc. of 10th East-European Conference on Advances in Databases and Information Systems (ADBIS 2006)*, Springer-Verlag, Thessaloniki, Greece, 2006.

[92] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., and Balakrishnan, H. Chord: a scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Transactions on Networking (TON)*, 11(1), ACM Press, NY, USA, pp. 17–32, 2003.

[93] Szymaniak, M., Pierre, G., and van Steen, M. Netairt: a DNS-based redirection system for apache. In *Proc. of International Conference WWW/Internet*, Algrave, Portugal, 2003.

[94] Tse, S. S. H. Approximate algorithms for document placement in distributed Web servers. *IEEE Transactions on Parallel and Distributed Systems*, 16(6), pp. 489–496, 2005.

[95] Vakali, A. and Pallis, G. Content delivery networks: status and trends. *IEEE Internet Computing*, 7(6), IEEE Computer Society, pp. 68–74, 2003.

[96] Valloppillil, V. and Ross, K. W. Cache array routing protocol v1.0. Internet Draft, 1998.

[97] Verma, D. C. *Content Distribution Networks: An Engineering Approach*, John Wiley & Sons, Inc., New York, 2002.

[98] Vixie, P. and Wessels, D. Hyper text caching protocol (HTCP/0.0). Internet Engineering Task Force RFC 2756, 2000. www.ietf.org/rfc/rfc2756.txt

[99] Wang, J. A survey of Web caching schemes for the Internet. *SIGCOMM Computer Communication Review*, 29(5), ACM Press, NY, USA, pp. 36–46, 1999.

[100] Wang, L., Pai, V. S., and Peterson, L. The effectiveness of request redirection on CDN robustness. In *Proc. of 5th Symposium on Operating Systems Design and Implementation*, Boston, MA, USA, pp. 345–360, 2002.

[101] Wessels, D. and Claffy, K. Internet cache protocol (ICP) version 2. Internet Engineering Task Force RFC 2186, 1997. www.ietf.org/rfc/rfc2186.txt

[102] Wu, B. and Kshemkalyani, A. D. Objective-optimal algorithms for long-term Web Prefetching. *IEEE Transactions on Computers*, 55(1), pp. 2–17, 2006.

第 3 章　动态、可扩展和高效的内容复制技术

Yan Chen

3.1 引言

处理器的性能、存储能力和网络带宽的指数性增长正在改变我们对计算的认识。我们的注意力已经从中心化的、手动配置的系统转到全球规模的分布式、自组织的复合体（由成千上万的组件组成）。然而，现有系统普遍可能存在着频繁的组件故障，以及出现因为网络连接缓慢或连接失败而导致容易断开的问题。因此，利用本地的资源非常重要——既为性能，也为可用性。进一步讲，普遍的流式应用必须精心设计它们的通信结构，以避免占用过多的资源。因此，要实现本地访问和有效的通信，需要在数据副本和组播节点的放置方面实现更大的灵活性。

为了在取得灵活、可扩展的同时保留数据本身的固有属性，一种方法是将系统划分为两个副本层[18]：一个小型、可靠的主层（primary tier）和一个大型、软状态（soft - state）的次层（second - tier）。主层可以代表一个 Web 服务器（用于 Web 内容分发）、一个存储系统的拜占庭内环[6,29]或一个流媒体服务提供商。主层的重要之处在于它必须保持数据的最新副本，并负责序列化和提交更新。我们将主层视为一个黑盒，简单地称之为“数据源”。次层变为软状态将是这一章谈论的重点。次层的例子包括内容分发网（CDN）、文件系统缓存或 Web 代理服务器缓存（proxy cache）。

因为次层副本是软状态，我们可以动态地增长和缩小它们的数量以满足系统约束。例如，我们可能希望在 QoS 上有一个基本保障，以确保每一个用户与其所访问的数据副本之间的最大网络延时有一个上界。因为数据副本会消耗资源，所以我们试图产生尽可能少的副本来尽可能地满足约束。因此，访问量高的数据可以存在数百乃至成千上万个副本，而访问量低的数据可能并不需要副本。

一个无约束副本所面临的难题是如何确保所存储的内容不会过期。某些稍为松弛的同步要求（如 Web[20]、OceanStore[29]或 Coda[26]），允许在数据源出现更新和更新传播之间出现一定程度的延时。然而，更新数据的发送仍然必须及时进行。大量潜在副本的存在使得进行直接的、点对点传送到副本的更新不可能实现。事实上，次层本身具有的流动特性表明副本自组织形成组播树的必要性，我们称这样的组播树为分发树（dissemination tree，d - 树）。由于内部节点必须将更新转发给子节点，所以将通过对树的扇出进行约束来设法控制这些内部节点的负载。

次层副本面临的挑战是：为用户提供良好的服务质量的同时，维持高效和均衡的设施资源消耗。为解决这一挑战，我们提出了一个自组织的软状态复制系统，称为可扩展内容接入网（Scalable Content Access Network，SCAN）。图 3.1 给出了一个 SCAN

系统的示意图。图中的网络层给出了两种类型的物理节点：SCAN 服务器（用正方形表示）和客户端（用圆圈表示）。假设 SCAN 服务器被放置在互联网中主要服务提供商（ISP）的互联网数据中心（IDC）内，具备与主干网的良好连接。每个 SCAN 服务器可能包含各种数据资源的副本。SCAN 系统的一个新特性在于它与一个分布式路由和定位（DOLR）系统（称为 Tapestry[22]）相结合。Tapestry 允许客户端在没有全局通信的情况下能够定位邻近的副本。

图 3.1 中展示了三种类型的数据：数据源和副本位于 SCAN 服务器中，是本章的讨论重点；缓存位于客户端，它们是数据的映像，不在本章的讨论范围内[⊖]。本章的目标是将客户端的数据请求变换为对副本的管理活动。本章的主要贡献为：

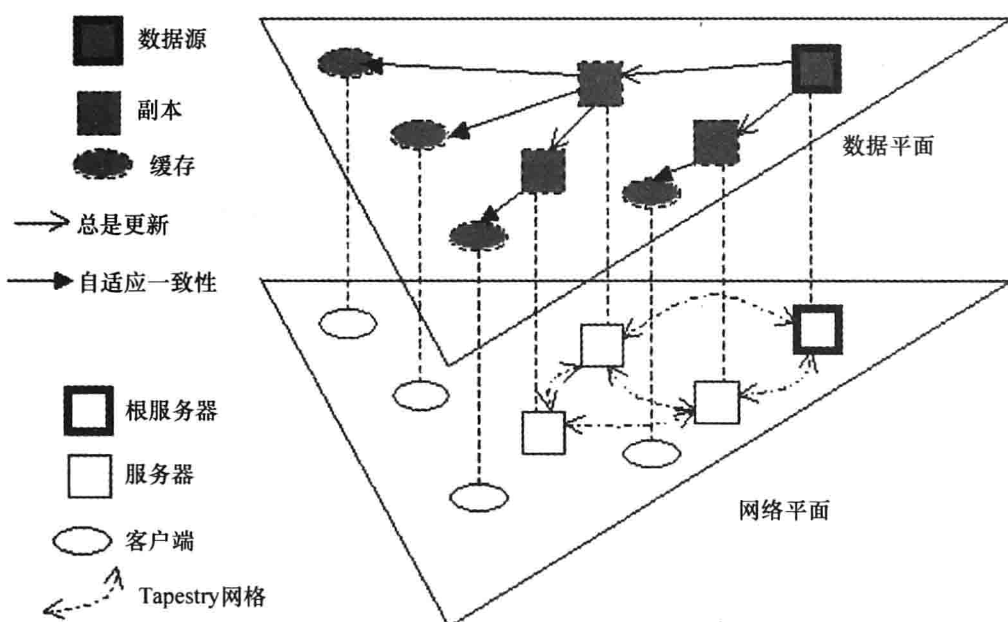


图 3.1 一个 SCAN 系统的架构

1）本章提供了一种算法，能够动态地放置最少数量的副本，同时满足客户端的 QoS 要求和服务器的容量约束。

2）这些副本能够以自组织的形式形成具有短延时和低带宽消耗的 d－树，以便用于更新的分发。

一个重要的直觉是，DOLR 系统的存在使得副本的同步放置和 d－树的构造不需要与数据源进行通信。因此，d－树的每个节点仅需为其父节点和子节点维持状态即可。

本章余下内容组织如下：在 3.2 节中介绍了该领域的相关工作；3.3 节中研究了副本放置问题；3.4 节介绍本章的算法；3.5 节讨论评估的方法；评估结果在 3.6 节中给出。

⊖ 有各种不同的方式保持缓存的一致性，如参考文献［44］。——原书注

3.2 前期工作

本节首先概要地介绍内容分发系统，包括 Web 缓存（3.2.1 节）、基于拉取的非协作式 CDN（3.2.2 节）、基于推送的协作式 CDN（3.2.3 节）。通过 SCAN 对这些系统进行了比较，比较结果在表 3.1 中做了总结。然后，针对 CDN 的三个主要部分讨论了以往的工作：目标定位服务（3.2.4 节）和组播技术更新分发（3.2.5 节）。最后，在 3.2.6 节中总结了前期工作的局限性。

表 3.1 不同互联网内容分发系统的比较

特性	Web 缓存（客户端创建）	Web 缓存（服务器创建）	基于拉取的非协作式 CDN	基于推送的协作式 CDN	SCAN
缓存副本共享（用于高效复制）	否，非协作式	是，协作式	否，非协作式	协作式	协作式
请求重定向的可扩展性	无重定向	一般，使用 Bloom 滤波器[15]来交换副本定位信息	不好，中心化 CDN 名服务器	不好，中心化 CDN 名服务器	好，离散化 DHT 定位服务
复制的粒度	每个 URL	每个 URL	每个 URL	每个站点	每个集群
分布式负载均衡	否	否	是	否	是
副本一致性	否	否	否	否	是
对容错性的网络监控	否	否	是，但是不可扩展的网络监控	否	是，可扩展式的网络监控

3.2.1 Web 缓存

缓存可以由客户端或服务器创建。面向广域和分布式的大多数系统都由客户端创建，如 Web 浏览器和 Web 代理服务器[32]。问题在于，这两种解决方案（Web 浏览器和 Web 代理服务器）都存在本质缺陷：因为客户端缓存无助于降低发往邻近计算机的流量，而且 Web 代理服务器也无法对相邻的代理服务器带来任何帮助。因此，缓存所起到的效用被完全限制在同一站点的客户端之间远程文档的低层共享[4]。一种可行的解决方法是在服务器端创建缓存，它允许服务器决定何时何地分发对象[3,4,21]。从本质上说，CDN（包括本章提到的方法）是由服务器来创建缓存，同时配备专用的边缘服务器。以前的由服务器创建的缓存系统都是基于一些不现实的假设。

Bestavros 等人将互联网建模为一个分层结构，其中任意内部节点均可作为服务的代理服务器（proxy）[3,4]。这种假设是不合实际的，因为内部节点实际上是路由器，不大可能作为服务的代理服务器来运行。地理意义上的推送缓存技术会基于对网络拓扑结构的全局知识和用户的访问模式来自行复制 HTML 网页[21]。最近，自适应 Web 缓存[34]和摘要缓存[15]等概念的提出，使得网站代理服务器之间的缓存共享成为可

能。缓存之间定期地交换内容的状态，可以消除延时和额外的缓存探测操作所带来的资源消耗。然而，每一个代理服务器需要将缓存内容的更新索引发送到所有其他的代理服务器，并需要存储其他代理服务器的内容索引。这样，即便有着像 Bloom 滤波器[15]那样紧凑的内容索引摘要，缓存服务器状态的维护和内容交换次数仍然非常可观，而且缓存服务器的数量要远远低于文件数量。例如，有研究表明，代理服务器的目标数量仅仅在 100 的数量级上[15]。此外，如果没有像 CDN 这样的专用设施，缓存代理服务器就无法应对网络拥塞或连接失败，也无法提供分布式负载均衡。

3.2.2 基于拉取的非协作式 CDN

近来，CDN 已经实现商业化，用于提供 Web 托管、互联网内容和流媒体传输等。基本上说，内容根据用户的要求被拉取到边缘服务器。目前，出现了各种机制将客户端请求定向到一台 CDN 服务器（又名 CDN 节点或边缘服务器），如基于 DNS 的重定向、URL 改写、HTTP 重定向等[1]。大部分的商业 CDN 提供商（如 Akamai[14]、LimeLight Networks[31]、Mirror Image[35]）为了实现透明性而使用基于 DNS 的重定向[28]。

图 3.2 显示了使用基于 DNS 重定向的 CDN 架构。鉴于 CDN 服务提供商的快速增长（例如，Akamai 已经在 69 个国家拥有超过 25 000 个服务器，这些服务器分布在大约 900 个网络[14]），可以假设对于每一个访问量较大的用户群，都为其配备一台 CDN 服务器以及一个本地 DNS 服务器。用户群是那些在拓扑结构上邻近的一组客户端。这些客户端可以通过他们的 BGP 前缀组合在一起[27]，也可以经由其本地 DNS 服务器聚在一起。后者相对简单，并且已在实践中得到了采纳，但不是很精确[33]。

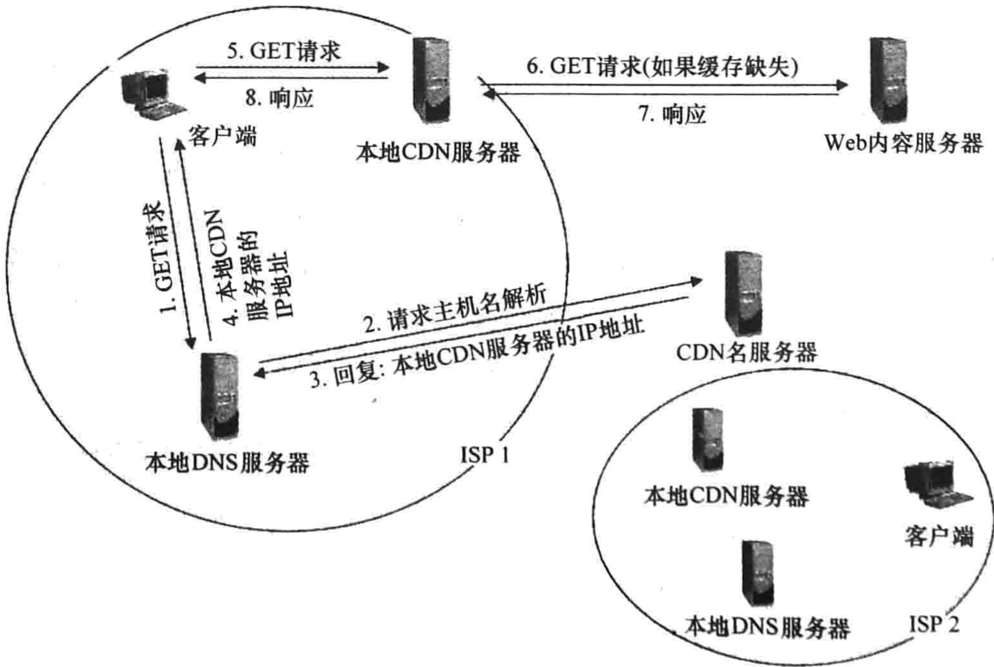


图 3.2 基于拉取的非协作式 CDN 架构

图3.2显示了客户端检索一个URL的操作顺序。首先，主机名解析请求经本地DNS服务器被送往CDN名称服务器。由于中心化定位服务的特性，CDN名称服务器没有能力保存所有网址副本的位置记录。因此，它只能根据网络邻近度、带宽可用性和服务器负载等信息进行请求的重定向。收到重定向请求的CDN服务器可能没有副本，这时它会从Web内容服务器上取副本内容，然后返回给客户端。

由于非协作方式的特性，目前的CDN往往需要放置更多的副本，因为存储和更新的关系CDN往往会消耗更多的资源。仿真结果显示，在保证合理的延时下，基于内容推送的协作式CDN（在3.2.3节中定义）仅仅使用了少量的副本服务器（6%～8%），更新分发带宽比非协作式方法小10%[10, 11]。

Rabinovich和Aggarwal提出了RaDaR，这是一个全球Web托管服务，可以实现动态内容复制和迁移[41]。不过，它需要DNS给出从客户端到服务器的完整路径，这在实际中往往无法实现。

3.2.3 基于推送的协作式CDN

最近的几项研究工作[25, 30, 40, 48]提出了一种新思路，即根据用户的访问模式和网络的全局拓扑结构，主动地将内容从源Web服务器推送到CDN边缘服务器或代理服务器，副本之间通过相互协作来满足用户的请求。

相对于传统方法而言，基于推送的协作式复制机制的主要优势并不在于使用推送而非拉取（只针对强制性失效㊀），而是来源于副本之间的协作共享。这种协作大大减少了副本的数量，从而降低了复制和更新的成本[10, 11]。

可以采用一个类似的CDN架构来支持这种基于推送的协同式内容分发，如图3.3所示。首先，基于内容的超链接结构和（或）访问历史（由CDN名服务器收集得到），Web内容服务器以增量的形式进行内容的推送[10, 11]。内容服务器在后台运行“推送”进程，将复制通知给CDN名称服务器，该服务器维护内容及其副本之间的映射（其中内容是通过改写后的URL中的主机名进行标识）。上述映射关系可以是粗粒度的（例如，如果以Web站点为单位进行复制，那么映射就实现在Web站点的层面上），也可以是细粒度的（例如，如果以URL为单位进行复制，那么映射就位于URL的层面上）。

有了这样的副本定位跟踪技术，CDN名称服务器就可以将用户请求重定向到与其最接近的副本。值得注意的是，基于DNS的重定向使地址解析能在每台主机上进行。我们将DNS重定向与内容修改相结合（如URL改写），以实现对每个对象的重定向[1]，也就是说，对不同对象的引用被改写为对不同主机的引用。为了降低域名空间的大小，可以对对象进行归并[10, 11]，归并后的每组对象共享同样的主机名，为此内容提供商可以在推送内容对象之前改写URL。因此，它不会影响客户端延时，并且这种一次性的系统开销是可以接受的。在这两种模式中，CDN的边缘服务器允许执行

㊀ 强制性失效（Compulsory miss）是指当第一次访问一个块时该块不在缓存中，需要从下一级存储器中调入缓存。——译者注

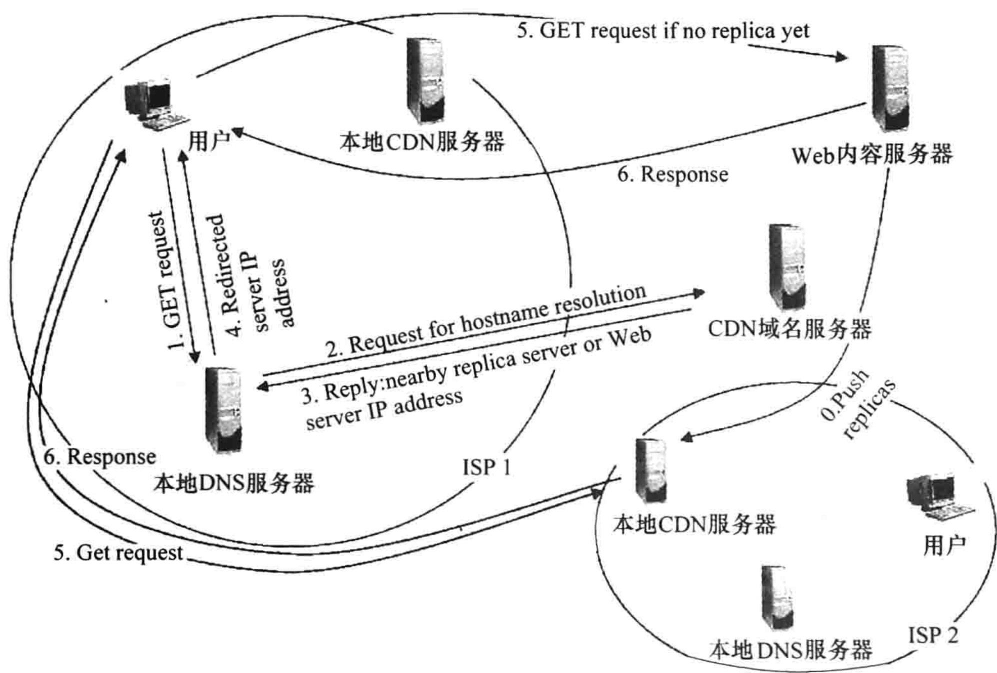


图 3.3　基于推送的协作式 CDN 架构

其缓存替换算法，也就是说，在基于推送的协作式内容复制中的映射关系是软状态。如果客户端在重定向后的 CDN 边缘服务器中无法找到内容，要么客户端可以向 CDN 域名服务器索要另一个副本，要么边缘服务器直接从 Web 服务器拉取内容并将其发至客户端。

在 Li 等人的方法中，处理代理服务器放置问题的方式是首先假设底层的网络拓扑结构是树，再将其建模成动态规划问题[30]。虽然第一步看上去很有新意，但这种方法面临一个重大的限制，即互联网的拓扑结构并不是树。最近的研究[25, 40]对实际的路由和拓扑结构进行评估，这些彼此独立的研究都表明，使用贪婪放置算法可以使 CDN 接近最优性能。

为了对已知全局网络拓扑结构和客户端分布这个前提进行简化，一种基于拓扑通知（topology – informed）的互联网副本放置方法在具有大扇出路由器的地方放置副本[42]。研究结果表明，使用基于路由器级拓扑的副本放置方法，所得到的平均用户延时比使用贪婪算法减少了 10% ~20%，但这需要对放置方法进行精心设计。

3.2.4 对象定位系统

网络化的应用已经将触角延伸到互联网的各种设备和服务。因此，当应用扩展到这些网络资源时，在广域尺度上进行对象的定位是一个重要问题。不仅如此，WWW 中广泛存在的以读为主的共享访问模式导致了大量的对象复制，这也带来了对象定位的问题。目前，很多方法用于解决定位服务的问题[13, 19, 23, 50]，大致可以分为以下三组：中心式目录服务（CDS）、复制式目录服务（RDS）和分布式目录服务（DDS）。

下面将详细讨论这些相关研究工作，不过据笔者所知，目前尚未出现对性能基准和性能比较的研究。

1. 中心式目录服务和复制式目录服务

一个中心式的目录服务驻留在单一服务器上，并提供网络上每一个对象的位置信息（见图3.4）。由于它驻留在单一服务器上，因此非常容易遭受DoS攻击。一个改进措施是使用复制式目录服务，它能够提供多个目录服务器。一个复制式目录服务具有更高的可用性，但保持数据一致性会引发额外的系统开销。本章并不考虑分区的目录服务，因为它经常需要额外的专用目录服务器来维护分区信息，如DNS的根服务器。

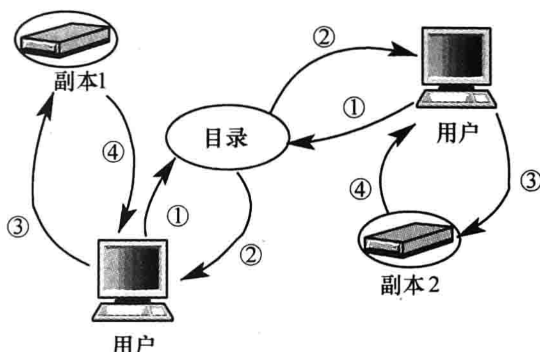


图3.4　一个中心式目录服务（CDS）：客户端访问单一目录服务器来得到一个邻近副本的位置信息，随后客户端直接访问副本。一个复制式目录服务（RDS）提供多个目录

2. 分布式目录服务：Tapestry

研究人员已经开始研究分散、对等和带有分布式哈希表（DHT）的定位服务，如CAN[43]、Chord[47]、Pastry[45]和Tapestry[50]等。这些服务提供一个分布式的网络设施，能保证对象的迅速定位。不同于依赖单一服务器定位对象，在这个方法中一个查询命令会在整个网络传播，直到一个节点知道所请求对象的位置信息。这种非中心化的定位服务能够提供非常高的可用性。在面对攻击时，即使一组节点由于受到攻击而瘫痪，所带来的影响也仅仅局限于很小的对象范围。

此外，Tapestry利用局部性将报文路由到移动端点（如对象的副本），这种方式不同于其他结构性对等覆盖网[43,45,47]。因此，使用Tapestry来建立SCAN系统。

Tapestry是一个IP覆盖网，它使用一个分布式和容错架构来跟踪网络中对象的位置。它有两个组成部分：路由网格和分布式定位服务。

（1）Tapestry路由网格

图3.5显示了Tapestry网的一部分。每个节点以分布式通过附近的代理服务器加入Tapestry网，建立与其他Tapestry节点的邻接连接（显示为实线箭头）。这些邻接连接提供了从每一个节点到所有其他节点的路由，路由过程每次解析一个目标地址中的一位（例如，＊＊＊8⇒＊＊98⇒＊598⇒4598，其中＊代表通配符）。这种路由方案是基于哈希后缀路由结构，最初由Plaxton等人[39]提出。

（2）Tapestry分布式定位服务

Tapestry给每一个对象分配一个全局唯一的Tapestry名称（GUID）。然后，将每一个GUID映射到唯一的根节点。存储服务器通过向根节点发送报文来发布对象，并在每一跳保存定位指针。图3.5显示了一个对象的两个副本和Tapestry根节点。这些映射仅仅是指向存有对象o的服务器s的指针，而非对象的副本。因此，对于附近的对象，用户的搜索报文可以很快与发布报文相遇，从而利用局部性实现快速搜索。有

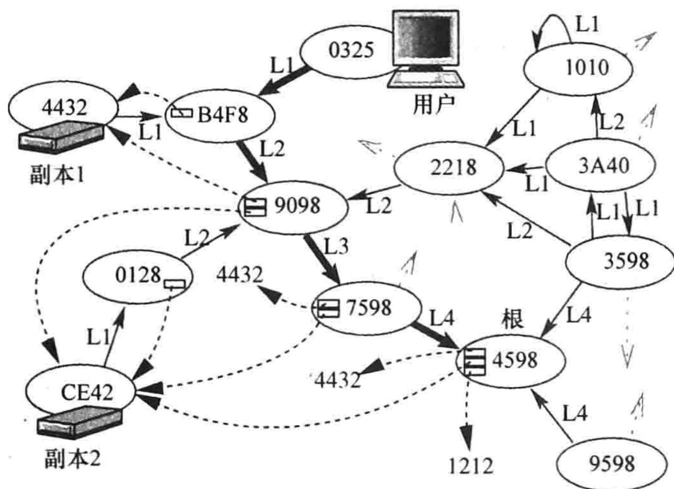


图 3.5 一个分布式目录（Tapestry）：节点通过连接（实线箭头）相连。节点之间的路由每次解析一位，如 1010 → 2218 → 9098 → 7598 → 4598。所有的对象与一个特定的“根”节点相关联（如 4598）。服务器发布副本时，向根节点发送报文，同时建立每个节点至副本的反向指针（虚线箭头）。客户端向根节点发送报文，当遇到指针后就可以直接路由到副本（如 0325 →B4F8 →4432）。

研究结果表明[39]，定位对象的平均距离与该对象的距离成正比。

3.2.5 分发更新的组播

对于更新的分发，如果将 IP 组播作为互联网内容分布的架构基础则存在根本性问题。例如，它在工作时只考虑空间因素，而不考虑时间，而互联网上的内容分发大部分都是同时基于两者[16]。此外，也没有广泛可用的域间 IP 组播。

作为替代方案，出现了许多应用层组播（简称 ALM）系统[7, 12, 16, 17, 38, 51]。其中一些[7, 12, 38]关注于小范围、多源的应用，如视频会议，而另一些[16, 17, 51]则专注于大尺度、单一来源的应用，如流媒体组播。Bayeus[51]也是建立在 Tapestry 之上，使用 Tapestry 位置服务来寻找组播根节点，然后使用 Tapestry 路由功能对控制报文（如“加入”）和数据报文进行路由。与之不同，我们只使用 Tapestry 的定位机制寻找附近的副本。

大多数 ALM 系统都有扩展性的问题，因为它们利用一个中心节点维持所有子节点的状态[7, 12, 17, 38]或处理所有“加入”的请求[51]。复制根节点是常见的解决方法[17, 51]，但其缺陷在于存在同步问题和通信成本的开销。另外，Scribe[46]和 SCAN 的更新组播系统（即分发树）利用点对点路由技术和定位服务解决了可扩展性问题。Scribe 是一个大尺度的事件通知系统，采用覆盖 DHT 进行订阅和传播。分发树更有效，因为使用覆盖 DHT 仅仅实现订阅，而使用 IP 可直接进行分发。

3.2.6 总结

综上所述，我们发现以往对 CDN 的研究和相关技术中存在以下局限性：

1）由客户端创建Web缓存是短视的做法，而由服务器创建Web缓存具有不可控的内容状态交换开销。这两种方法都不能适应网络拥塞和失效，也不能提供分布式负载均衡。

2）CDN依赖于中心化的定位服务，因此必须要么使用低效的基于拉取的复制方法（非协作式CDN），要么以网站的粒度进行复制，同时牺牲了客户端性能（协作式CDN）。

3）没有为现有定位服务提供性能基准或DoS攻击恢复度基准，因此难与其他方法进行比较。

4）副本/缓存没有一致性机制，互联网中不存在IP组播，而现有的应用层组播则存在着扩展性问题。

在SCAN中，前两个局限性可以通过分布式定位服务（即Tapestry）得到解决，本章也提出了一个网络DoS恢复度基准用来与其他方法进行性能对比[8]。对于第四种局限性，本章的解决方法是动态放置副本并将其自组织成一个可扩展的应用层组播树来分发更新，这将在下面进行讨论。

3.3 动态副本放置问题

如图3.1所示，副本放置是SCAN中的一个关键组成部分。根据用户的要求，它动态地放置最小数量的副本来同时满足QoS和服务器容量的限制。上节中所讨论的定位服务通过Tapestry的PUBLISHOBJECT API函数获取新副本的信息[50]。

将网页副本放置问题转化为一个优化问题并对其建模，则需要考虑很多因素，下面对该问题进行描述。考虑一个流行度很高的Web站点或CDN的托管服务器，其目的是通过将内容推送到若干托管服务器节点来提升自身性能。问题是需要动态决定内容在哪里被复制，从而使得一些相应的目标函数能够在动态流量模式和用户的QoS约束和（或）资源的约束下得到最优解。这个目标函数要么能够最大限度地降低延时、丢包率、吞吐量等QoS指标，要么可以最小化CDN服务提供商的复制成本（如网络带宽消耗）或一个整体成本函数（如果每个连接都被赋予一个成本）。对于Web内容的分发，复制成本中的主要资源消耗是每个接入主干网的互联网数据中心（IDC）的网络访问带宽。因此，对于一个给定的Web对象，成本和副本数量呈线性正比关系。

在Qiu等人的研究中，试图在副本数量的约束下最小化所有用户请求的总响应延时[40]。我们从另一个角度来分析这个副本放置问题，即最小化副本数量的同时，满足用户对于延时的约束和服务器能力的约束。在这里，假设用户能够根据与CDN提供商协商达成的服务水平协定（SLA）提供一个合理的延时约束。这样，定义Web内容放置问题如下：

假设网络G由客户端集合C和服务器节点集合S构成，每个用户c_i都有其延时约束d_i，并且每个服务器s_j都有其负载/带宽/存储容量的限制l_j。现在的问题是找到一个最小的服务器集合S'，使得任意用户c_i和其“父”服务器$s_{c_i} \in S'$之间的距离有界

（由 d_i控制）。更正式的表述为：找到最小的 K，使得存在集合 $S' \subset S$，其中 $|S'| = K$ 且对于 $\forall c \in C \exists s_c \in S'$有 $\text{distance}(c, s_c) \leq d_c$。与此同时，这些客户端 C 和服务器 S' 自组织成一个应用层的组播树，其中 C 作为叶节点，且对于 $\forall s_i \in S'$，其扇出度（即直接子节点数）满足 $f(s_i) \leq l_i$。

3.4 副本放置算法

使用基于 DOLR 并结合路由局部性算法，可以同步实现副本放置和树的构建。每一个 SCAN 服务器都是 DOLR 的成员，这样新的副本就被发布到 DOLR 中。此外，每个客户端将其请求定向到其代理 SCAN 服务器，此代理服务器与其他 SCAN 服务器进行交互，并将内容分发到客户端。

尽管在构建树的过程中使用 DOLR 来定位副本，但通信是通过 IP 进行的。特别地，在 d－树的节点间使用 IP，以使父节点和子节点之间保持跟踪。此外，如果客户端发出的请求导致一个新副本的放置，那么客户端的代理服务器会保存一个指向该副本的缓存指针，这就可以在代理服务器到副本之间直接对请求进行路由。缓存指针是软状态的，因为一直可以使用 DOLR 来定位副本。

3.4.1 副本放置的目标

副本放置的目的在于同时满足客户端延时和服务器负载的约束。客户端延时是指一个客户端从 SCAN 系统读取信息所需要的往返时间，我们设法将其保持在预先定义的约束范围内。服务器负载是指给定服务器需要处理的通信量，假设该负载与服务器服务的用户数和 d－树的子节点数直接相关，并设法使其低于一个特定的上限值。我们的目标是在满足这些约束的同时，最小化所部署的副本数量，同时保持 d－树的平衡以及在更新时产生尽可能少的流量。在 3.6 节中将给出实验结果。

3.4.2 动态放置

我们的动态放置算法分为副本搜索和副本定位两个阶段。副本搜索试图找到已有的副本，并在不过载的情况下满足延时约束。如果这一步成功，则在客户端放置一个链接，并将其缓存在客户端的代理服务器上。如果没有成功，则进入副本放置阶段添加一个新的副本。

在副本搜索阶段，使用 DOLR 寻找与客户端代理服务器较近的副本，称为“入口”副本。DOLR 的局部性能够确保该入口副本可以作为一个合理的备选副本与客户端进行通信。此外，由于 d－树已经联通，因此此入口副本可以与所有其他副本交互。这样，我们可以设想存在三种搜索方式：单一的（Singular，只考虑入口副本）、局部的（Localized，考虑到父节点、子节点和同级节点的入口副本）及穷举的（Exhaustive，考虑所有副本）。对于一个给定的方式，我们检查所有的相关副本并在其中选择一个满足约束的副本；如果没有任何副本满足约束，则建立一个新的副本。

```
procedure    DynamicReplicaPlacement_Naive(c,o)
1  c 通过 DOLR 向 o 发出 JOIN 请求，到达入口服务器 s。采集路径中每个服务
   器 s'的 IP_s'、dist_overlay（c, s'）和 rc_s'。
2  if rc_s > 0then
        if dist_overlay（c, s）≤d_c  thens 成为 c 的父节点，exit。
        else
3           s 对 c 运行 ping 命令，得到 dist_IP（c, s）。
4           if dist_IP（c, s）≤d_c  then s 成为 c 的父节点，exit。
        end
   end
5  在 s 处，选择路径中满足 rc_s' > 0 且 dist_overlay（t, c）≤d_c最小的 s'。
   if 不存在这样的 s'  then
6           对路径中的每一个服务器 s'，s 计算 dist_IP（c, s'），选择满足条件
            rc_s' > 0 且 dist_IP（t, c）≤d_c的使 dist_IP最小的 s'。
   end
7  s 在 s'处放置一个副本，并作为其父节点，s'作为 c 的父节点。
8  s'在 DOLR 中发布副本，exit。
```

算法1：简单动态副本放置。符号定义：对象o，延时约束为d_c的用户c，入口服务器s，每一个服务器s'的剩余容量为$rc_{s'}$（服务器能处理的额外子节点），覆盖距离（$dist_{overlay}$（x, y））和IP距离（$dist_{IP}$（x, y））分别表示在覆盖网和IP网上的往返时间（RTT）。

从用户代理的服务器发送报文到入口副本时，将副本的放置限制在被DOLR路由协议访问的服务器。我们能够在没有全局IP拓扑先验知识的情况下也能定位这些服务器。DOLR的局部特性表明这些地点都适合副本的放置。本章考虑两种放置策略：Eager方法将副本放置在距离用户尽可能近的地方；Lazy方法将副本放置在距离用户尽可能远的地方。如果所有满足延时约束的服务器都过载，就替换一个旧的副本。如果入口服务器过载，则断开其d-树中建立时间最久的连接。

```
procedure    DynamicReplicaPlacement_Smart(c,o)
1  c 通过 DOLR 发送 JOIN 请求到 o，到达入口服务器 s。
2  请求从 s 传递到子节点（sc）、父节点（p）和同级节点（ss）。
3  每一个 rc_t > 0 的家庭成员 t 发送 rc_t 到 c，c 通过 TCP 的三次握手来测
量dist_IP（t, c）。
4  if  存在 t 且 dist_IP（t, c）≤ d_c  then
5       c 选择 rc_t值最大且 dist_IP（t, c）≤d_c的 t 作为父节点，exit。
   else
```

```
6    c 通过 DOLR 发送 PLACEMENT 请求到 o，到达入口服务器 s。
     采集路径中每个服务器 s' 的 IP_s'、dist_overlay（c，s'）和 rc_s'。
7    在 s 处，选择路径中满足 rc_s' >0 且 dist_overlay（t，c）≤d_c 最小的 s'。
     if  不存在这样的 s'  then
8        对路径中的每一个服务器 s'，s 计算 dist_IP（c，s'），选择满足 rc_s' >0
         且 dist_IP（t，c）≤d_c 条件的使 dist_IP 最大的 s'。
     end
9    s 在 s' 处放置一个副本，并作为其父节点，s' 作为 c 的父节点。
10   s' 在 DOLR 中发布副本，exit。
end
```

算法2：智能动态副本放置。符号定义：对象 o，延时约束为 d_c 的用户 c，入口服务器 s，每一个服务器 s' 的剩余容量为 $rc_{s'}$（服务器能处理的额外子节点），覆盖距离（$dist_{overlay}$（x，y））和 IP 距离（$dist_{IP}$（x，y））分别表示在覆盖网和 IP 网上的往返时间（RTT）。

1. 动态技术

现在可以将上述搜索和放置方法进行组合，生成动态副本管理算法。下面重点介绍两种组合：

（1）简单放置（Naive Placement）

这是单一搜索法和 Eager 放置法的一个简单组合，这种启发式方法可以使搜索和放置的流量最小。

（2）智能放置（Smart Placement）

这是一个较为复杂的算法，如算法 2 所示。该算法利用局部搜索法和 Lazy 放置法。

注意，我们试图使用覆盖延时来估计 IP 延时，以便减少 ping 报文的数量。在这里，用户可以在打开浏览器时启动一个由 CDN 服务提供商提供的后台程序，以便能主动地参与到这些协议中。Tapestry 网的局部特性自然导致了 d－树的局部性，即子节点和父节点之间倾向于彼此接近（用 IP 跳数来衡量），这就可以在分发更新时降低延时和组播带宽的消耗（测量结果见 3.6 节）。与简单放置方法相比，智能放置方法消耗了较多的“加入”流量来构造了一个具有较少副本、覆盖更多客户端、具有更少的延时和组播带宽消耗的树。我们将在 3.6 节评估这两种方法的权衡效果。

2. 静态比较

我们知道，上面给出的副本放置方法在副本的部署数量上不太可能达到最优，这是因为客户端是逐个地加入，并且对网络拓扑结构所知有限。在静态方法中，根服务器具备对网络拓扑的完整信息，并在得到所有来自客户端的请求后才放置副本，更新则是通过 IP 组播进行分发。静态放置方法不太符合实际情况，但是因为它可以利用用户分布情况和网络全局拓扑结构，因此反而能提供更好的性能。

3.3节中定义的问题可以转换为限量设施定位问题的一种特殊形式[24]，其定义为：给定一组构建设施的位置 i，在位置 i 的设施构建成本为 f_i，每一个客户端 j 必须被分配到一个设施，由此引入的成本为 $d_j c_{ij}$，其中 d_j 表示节点 j 的设施需求，c_{ij} 表示从 i 到 j 的距离。每一个设施最多能够服务 l_i 个客户端，因此目标就是找到使整体成本最小的设施数量和位置。

为了将设施定位问题转换为我们要求的形式，设置代价 f_i 恒为1，当位置 i 能够覆盖客户端 j 时设 c_{ij} 为0，否则设 c_{ij} 为∞。目前已知最好的近似算法使用原始对偶算法和拉格朗日松弛算子，确保解的上界不高于最优解的4倍[24]。然而，这种算法对于实际应用而言显得过于复杂。所以，我们设计了一个贪婪算法，该算法具有对数级的近似比率。

除了上面的符号，我们定义以下变量：C_s 表示被 s 覆盖的客户端集合，其中 $C_s \in C$ 且对于 $\forall c \in C_s$ 有 $dist_{IP}(c, s) \leq d_c$；S_c 表示客户端 c 可能的服务器父节点集合，其中 $S_c \in S$ 且对于 $\forall s \in S_c$ 有 $dist_{IP}(c, s) \leq d_c$。

```
procedure ReplicaPlacement_Greedy_DistLoadBalancing(C,S)

输入：被覆盖的客户端集合：C，所有服务器的集合：S
输出：被选择用于副本放置的服务器集合：S'

while C 非空 do
    找到 s∈S，其中 s 具有 min（|C_s|，rc_s）的最大值
    选择 min（|C_s|，rc_s）中的最大者 s∈S
    S' = S' ∪ {s}
    S = S - {s}
    if  |C_s| ≤ rc_s then   C = C - C_s
    else
        以|S_c|为关键字对 C_s 中的每个元素 s 进行升序排序
        选择 C_s 中最前面的 rc_s 个客户端作为 C_sChosen
        C = C - C_sChosen
    end
    对∀c∈C，重新计算 S_c
end
return S'
```

算法3：带负载均衡的静态副本放置。

现在考虑两种类型的静态副本放置方法：

1）IP静态。根节点具有全局IP拓扑的知识。

2）覆盖静态。对于每一个客户端 c，根节点只知道从 c 到根节点，且覆盖该客户端的Tapestry路径上的服务器（以IP距离计算）。

前者是一种“确保不超过”的最佳放置，我们期望它产生的副本总数最少且组播流量最低；后者使用了我们能想到的最好方法来从 DOLR 系统收集所有的拓扑信息。

3.4.3 软状态树管理

软状态的设施有可能具有极大的鲁棒性，这是因为它们可以很容易地通过重新配置来适应各种环境。SCAN 系统的目标是实现两种改进：故障恢复和性能调优。

为了实现故障恢复，数据源通过 d－树周期性地发送“心跳”报文。成员服务器知道心跳报文的频率，所以当它们在足够长的时间内还没有收到心跳报文时就会采取相应的措施。与上面提到的副本搜索阶段类似，这时副本会启动一个“重新加入”的进程来寻找新的父节点。此外，每个成员服务器定期向其父节点发送“刷新”报文。如果父节点在一定的预设时间内没有收到刷新消息，则认为与子节点的连接中断。有了这种软状态分组管理，任何 SCAN 服务器的崩溃都不会对 CDN 的总体性能产生显著的影响。

性能调优包括删除节点和重新平衡化 d－树。如果客户端的流量较低，则叶节点上的副本将被去除。为了平衡 d－树，每个成员服务器定期地重新加入以寻找新的最佳父节点。

3.5 评估方法

因为 ns2[5] 的规模只能达到 1000 个节点，所以本节为 SCAN 实现了一个基于事件驱动的模拟器，包括一个数据包级的网络模拟器（使用 Tapestry DOLR 的一个静态版本）和一个副本管理框架。软状态副本层通过模拟的客户端工作负载进行驱动。本节的评估方法包括评价指标、网络设置和工作负载。

3.5.1 评价指标

我们的目标是对 3.4.2 节中提到的副本策略进行评估，这些策略分别是动态简单放置（od_naive）、动态智能放置（od_smart）、覆盖静态放置（overlay_s）和 IP 静态放置（IP_s）。我们通过三类指标对这四种策略的效果进行比较：

（1）副本放置的质量

包括副本的部署数量和负载分布度，后者等于每一个副本服务器下面的客户端数量均值与标准差的比。

（2）组播性能

我们测量相对延时惩罚（RDP）和带宽消耗，其中带宽消耗通过对网络中所有连接的传输时间与字节数的乘积求和得到。例如，假设两个连接的延时分别为 10ms 和 20ms，则它们传输 1KB 的带宽消耗为 1KB × (10 + 20) ms = 0.03(KB · s)。

（3）树的构建流量

该指标包括应用层发送的报文数与用于部署副本和构建 d－树所消耗的带宽。

此外，我们还通过计算在受约束或不受约束时的最大负载来对容量约束的效果进

行量化，其中最大负载定义为所有 SCAN 服务器上的客户端缓存子节点的最大数量。另外，还针对各种客户端/服务器的比率和服务器的密度进行了敏感性分析。

3.5.2 网络设置

为了评估复制策略，使用 GT – ITM transit – stub 模型生成 5 个包含 5000 个节点的拓扑结构[49]，并对这 5 个拓扑结构上的实验结果进行平均。一个数据包级、基于优先级队列的事件管理器被用于仿真网络延时。该模拟器对物理链路的分发延时进行建模，但忽略了带宽限制、排队延时和包丢失的因素。

我们利用两个策略来放置 SCAN 服务器，其中一个策略是随机选择所有的 SCAN 服务器（记为随机 SCAN），另一个则是优先选择传输节点（transit node）和网关节点（记为主干 SCAN）。

为了与基于 DNS 重定向的 CDN 进行比较，我们模拟了该系统所具有的典型行为。在实验中，假定每一个客户端请求都被重定向到最近的 CDN 服务器上，这些服务器将为客户端缓存被请求信息的副本。这意味着，流行度高的对象有可能被缓存在每个 CDN 服务器上。另外，假设内容服务器可以通过 IP 组播将更新数据发送到副本服务器上。

3.5.3 工作负载

为了评估复制方案，我们不仅使用了一个人工生成的工作负载，和 Web 服务器收集的真实访问日志，这些工作负载是探索 SCAN 更广泛应用的第一步。

人工生成的工作负载是对瞬时拥塞简化后的近似。瞬时拥塞是一种不可预测的、由事件驱动的流量激增，会拖垮服务器并终止网站的服务。在仿真中，所有的客户端（而非服务器）随机向某一给定热门对象发出请求。

实验中基于跟踪驱动（trace – driven）的仿真包括一个大型的热门商业新闻网站，即 MSNBC[36]，以及来自美国宇航局肯尼迪航天中心[37]的跟踪记录。

表 3.2 显示了详细的跟踪信息。我们通过以下方式使用访问日志：使用从 BBN-Planet（Genuity）路由器中获取的 BGP 表[2]，基于 BGP 前缀对 Web 客户端进行分组[27]。对于美国宇航局的跟踪记录，因为在跟踪中大多数入口包含主机名，所以我们基于域名对客户端进行分组，其中域名定义为主机名的最后两项。例如，a1. b1. com 和 a2. b1. com 属于同一个域。假设可以模拟的最大拓扑结构为 5000 个节点（受限于内存容量），我们模拟了 MSNBC 的 4000 个顶级客户端组（覆盖了请求总数的 86.1%）和美国宇航局的所有客户端组。由于这些客户端不可能在传输节点或服务器节点，所以将之随机映射到拓扑结构中的其余节点。

表 3.2　Web 站点访问记录统计

网站	时段	请求数		客户端总数	客户端组数		仿真的对象
		全部	仿真		全部	仿真	
MSNBC	10 – 11 am，8/2/99	1604944	1377620	139890	16369	4000	4186
NASA	全天，7/1/95	64398	64398	5177	1842	1842	3258

3.6 评估结果

在本节中，我们评估了 SCAN 动态副本管理算法的性能，结果显示：

1）对于真实的工作负载，相对于 IP 组播的静态副本放置，SCAN 的副本放置数量接近最优值，同时能够提供良好的负载均衡、低延时以及合理的更新带宽消耗。

2）无论复制还是更新带宽消耗，SCAN 都优于现有的基于 DNS 重定向的 CDN。

3）SCAN 的性能对 SCAN 服务器放置、客户端/服务器比率以及服务器密度等因素相对不敏感。

4）容量约束对于负载均衡非常有效。

我们将先给出对人工负载的仿真结果，然后再提供实际 Web 跟踪结果。

3.6.1 人工工作负载的结果

我们首先对人工模拟的瞬时拥塞负载进行分析。SCAN 服务器通过“随机”或“主干”的方式选择 500 个节点，其余的节点作为客户端以随机的顺序访问一些热门对象。我们随机选择一个非传输节点的 SCAN 服务器作为数据源，并把热门对象的大小设为 50KB。此外，假设延时约束为 50ms，负载能力设为每个服务器处理 200 个客户端。

1. 策略之间的比较

图 3.6 显示了放置副本的部署数量和这些服务器上的负载分布。od_smart 使用的服务器数量仅占 od_naive 使用量的 30% ~60%，甚至好于 overlay_s，非常接近最优的 IP_s。另外需要注意的是，无论随机 SCAN 还是骨干 SCAN，od_smart 的负载分布比 od_naive 和 overlay_s 都要好，接近了 IP_s。

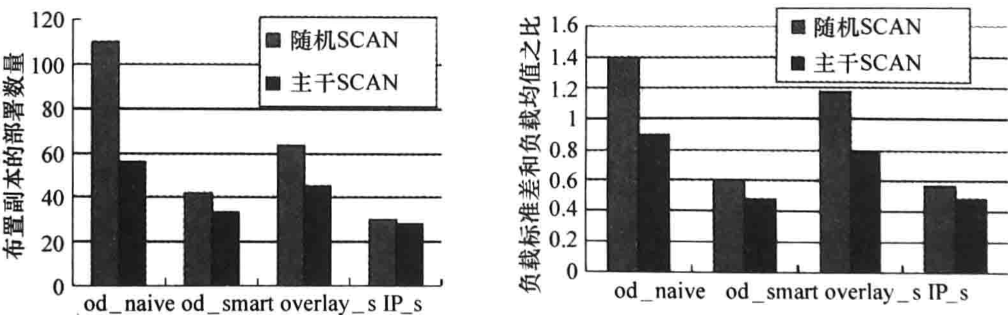


图 3.6 部署的副本数量（左图）和被选服务器的负载分布（右图）（500 个 SCAN 服务器）

相对延时惩罚是 d－树中任一成员与根节点之间的覆盖延时和单播延时的比率[12]。图 3.7 显示，od_smart 的 RDP 指标比 od_naive 要好，其中 85% 的 RDP 值都在 4 以内。在图 3.8 中，对各种副本放置技术与最佳的 IP 静态放置方法的带宽消耗进行了对比，结果是非常令人鼓舞的：od_smart 的带宽消耗非常接近 IP_s，并且比 od_naive 小得多。

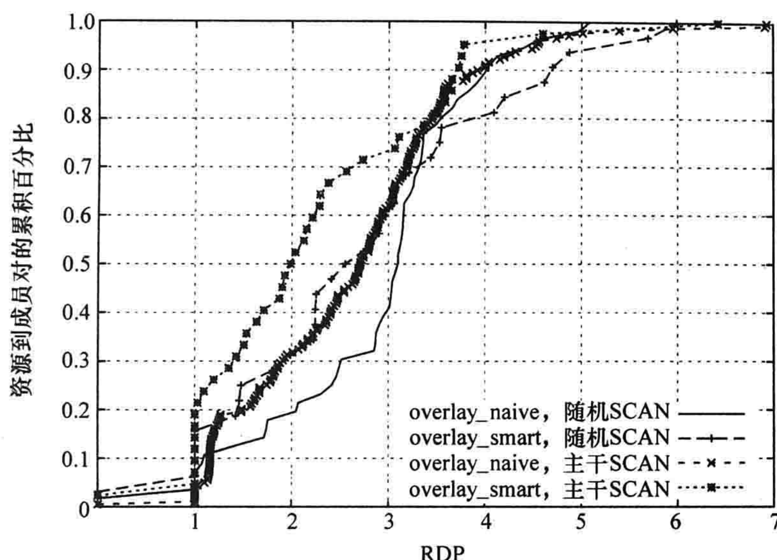


图 3.7 RDP 的累积分布（500 个 SCAN 服务器）

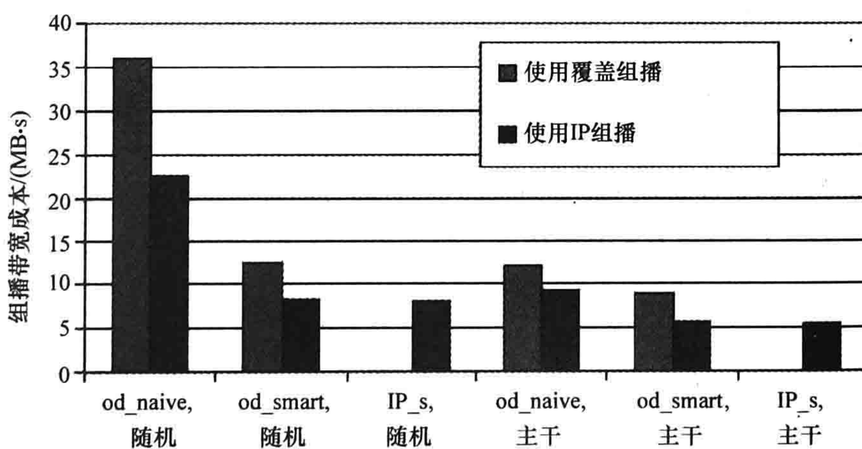


图 3.8 1MB 更新组播的带宽消耗（500 个 SCAN 服务器）

上述性能的取得都是建立在一定的 d－树构建成本之上（见图 3.9）。然而，无论随机还是主干 SCAN，od_smart 产生的报文数比 od_naive 少 3 倍，是最优情形 IP_s 的 1/6。

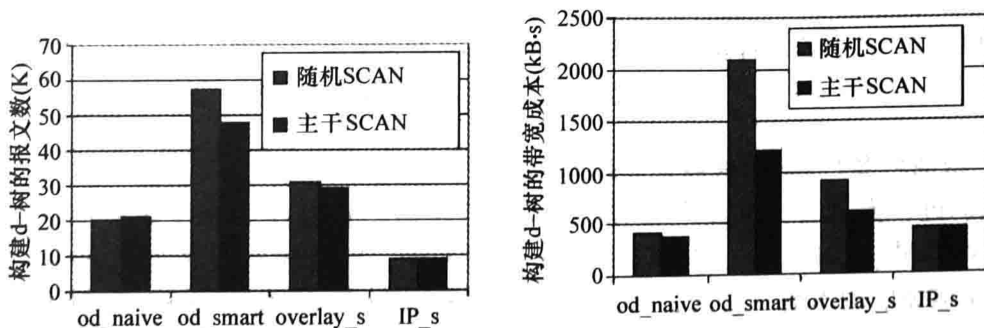


图 3.9 构建 d－树所需要的应用层报文数（左图）和总带宽消耗（右图）（500 个 SCAN 服务器）

同时，od_naive 使用了与 IP_s 几乎相同的带宽，而 od_smart 使用的带宽是 IP_s 的 3~5 倍。

总之，智能动态算法的性能接近理想情况（使用 IP 组播的静态放置）。与简单动态算法相比，它可以实现接近最优的副本数量、更好的负载分配以及更少延时和组播带宽占用。当然，代价是 3~5 倍于简单动态算法的 d-树构建流量。然而，由于 d-树构建的频率要远远低于数据访问和更新，因此从总体性能上看这是一个很好的折中。

由于服务器的分布和（或）数量有限，在面临 QoS 和容量需求时有可能一些客户端不能被覆盖到。在这种情况下，我们的算法可以有助于决定在哪里放置更多的服务器。实验结果表明，由于很难做到负载均衡，简单动态方法未覆盖到的客户端数量要远远多于智能方法，因此在下面对人工负载的研究中将不再考虑简单方法。

2. 与 CDN 的比较

同时，我们也将覆盖智能方法与基于 DNS 重定向的 CDN 进行了比较。结果表明，相对于传统的 CDN 而言，在分发更新时覆盖智能方法使用的副本数仅占传统 CDN 的 6%~8%，带宽消耗小于 10%。

3. 分布式负载均衡的有效性

下面将验证容量约束如何对三个客户端组（100、1000 和 4500）实现负载均衡，其中前两个是从 4500 个客户端中随机抽取生成。图 3.10 显示，缺乏容量约束（记为 w/o LB）会导致热点或拥塞情况的出现：一些服务器需要承担的负载量达到了其最高能力的大约 2~13 倍。为了方便比较，将使用负载均衡后的性能记为 w/LB。

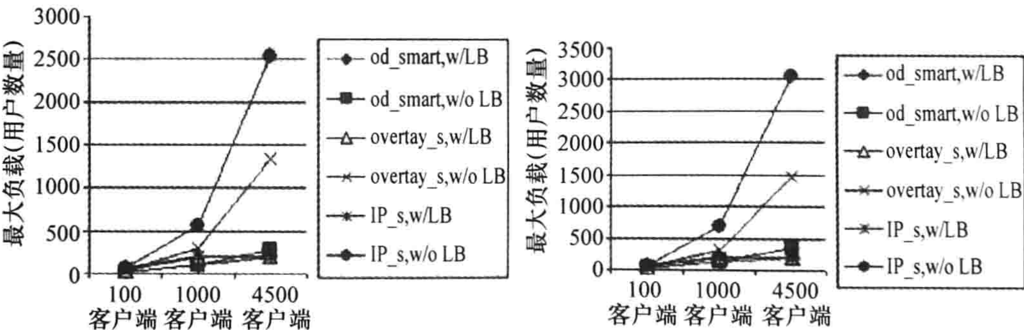


图 3.10　对于不同数量的客户端，使用和未使用负载均衡约束（LB）所测量到的最大负载（左图：500 个随机服务器，右图：500 个主干服务器）

4. 性能对客户端/服务器比率的敏感性

我们进一步使用三组客户端对 SCAN 进行评估。图 3.11 显示了副本部署的数量。当客户端数量较少时，w/LB 和 w/o LB 区别不大，因为这时没有服务器超过限制。od_smart需要的副本数一直低于 overlay_s，并保持在 IP_s 的 1.5 倍以内。与前面类似，我们也仿真了其他的性能指标，包括不同客户端/服务器比率下更新组播的负载分布、延时和带宽惩罚等。得到的结果大体相似，即 od_smart 总是好于 overlay_s，且非常接近于 IP_s。

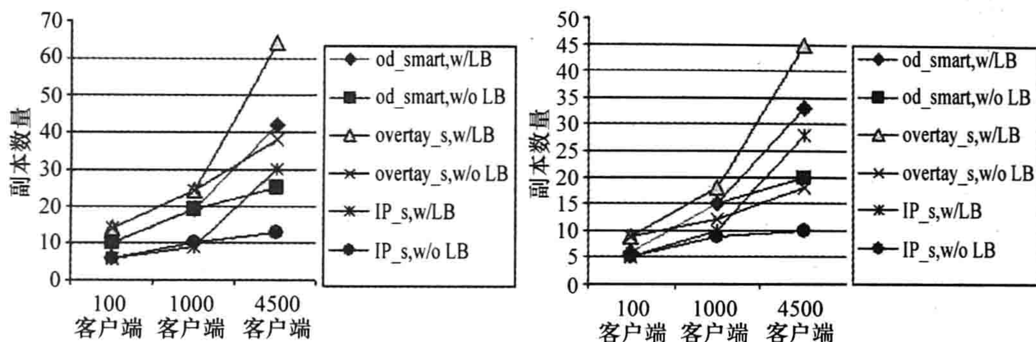


图3.11 对于不同数量的客户端，使用和未使用负载均衡约束（LB）所测量到的副本部署数量（左图：500个随机服务器，右图：500个主干服务器）

5. 性能对服务器密度的敏感性

接下来，我们提高了SCAN服务器的密度，从5000个节点中随机选择2500个作为SCAN服务器。结果表明，这种设置可以为客户端提供更好的QoS，同时也需要更少的服务器容量。这样，我们将延时约束设置为30ms，容量约束为50个客户端/服务器，客户端的数量在100~2500之间变化。

在SCAN服务器密度很高的情况下，od_smart使用的副本数仍比overlay_s少，尽管两者的值相当接近。IP_s仅需要一半左右的副本，如图3.12所示。此外，我们注意到，负载均衡仍然发挥着作用，也就是说，不对算法进行精心设计而仅仅简单地投入更多的服务器，无助于避免服务器过载或拥塞问题。

总之，在不同的SCAN服务器部署、客户端/服务器比率和服务器密度的配置下，od_smart均表现良好，说明基于容量约束的分布式负载均衡是有效的。

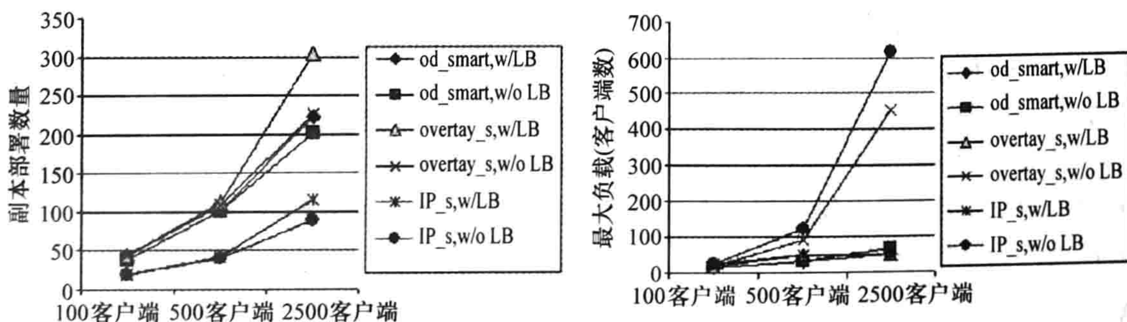


图3.12 使用2500个随机SCAN服务器，使用和未使用负载均衡约束（LB）所测量到的副本部署数量（左图）和最大负载（右图）

3.6.2 Web跟踪的工作负载情况

下面研究SCAN在对热门度差异很大的不同文档进行Web跟踪时的性能。图3.13a给出了我们使用的两个跟踪的请求分布（需要注意的是x轴是对数坐标）。结果表明，对不同URL的请求数在这两个跟踪上的分布非常不均匀。

对跟踪中的每个 URL，我们计算了 od_naive、od_smart 和 IP_s 所产生的副本数量。随后，对 od_naive 和 od_smart 产生的副本数量进行归一化，也就是将这两个数量与 IP_s 产生的副本数量相除。图 3.13b 中给出了 NASA 和 MSNBC 的这一比率的累计分布函数（CDF）。从图中可以发现，各条 CDF 曲线中百分比较低的部分相互重叠，并接近 1。产生这个现象的原因在于大多数 URL 很少有人访问，而且我们仅仅仿真了一个有限的时间，因此三种方法所得到的被部署的副本数量较少，而且彼此相差不大。然而，对于热门对象而言，od_smart 和 od_naive 的差异很大，对应于图中百分比较高的部分。对于所有对象，od_smart 非常接近于 IP_s，NASA 的比率不到 2.7，MSNBC 大约是 4.1，而使用 od_naive 时上述两个比率分别达到 5.0 和 15.0。

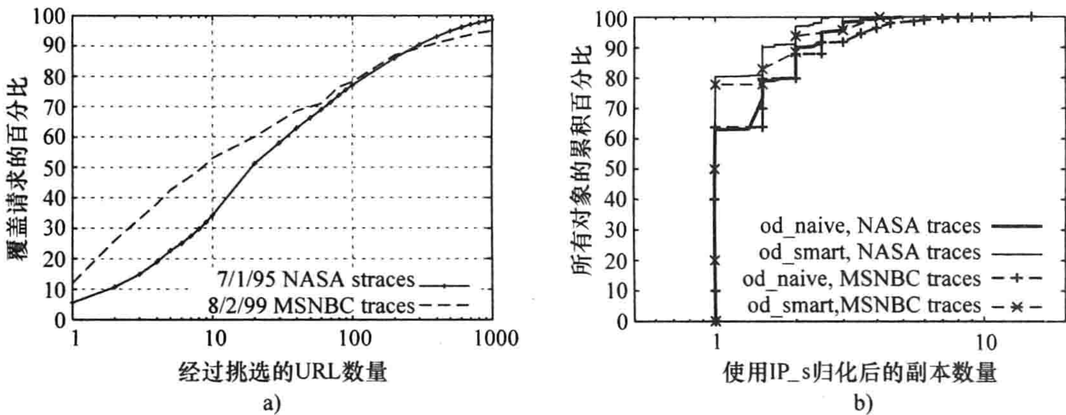


图 3.13　在 100 个主干 SCAN 服务器上对 NASA 和 MSNBC 的跟踪仿真

a）选择不同数量的顶级 URL 时所覆盖的请求百分比　b）使用 od_naive 和 od_smart 产生的副本数量的 CDF（这两个数量使用 IP_s 产生的副本数量进行了归一化）

此外，我们对分发更新时的带宽消耗进行了对比。给定一个单位大小的更新，我们使用以下三种方式分别计算了所有 URL 的带宽消耗：①od_naive 树的覆盖组播；②od_smart 树的覆盖组播；③IP_s 树的 IP 组播。与上面一样，使用③的结果对①和②的结果进行归一化，得到的 CDF 曲线与图 3.13b 非常相似。

从上面的分析可以发现，对访问不太频繁的对象或冷门对象来说，od_smart 和 od_naive的性能表现比较相似，不过对于热门对象（占据了主要的访问量），od_smart 的性能要明显好于 od_naive。

3.6.3　讨论

Tapestry 中拓扑结构的改变是如何影响副本放置的？需要注意的是，Tapestry 中的覆盖距离一般来说要比 IP 距离大 2～3 倍。3.6 节的仿真结果显示了由此带来的不利结果：在副本放置中 overlay_s 与 IP_s 使用了完全相同的算法，但 overlay_s 使用静态 Tapestry 级的拓扑而不是 IP 级拓扑。仿真结果表明，overlay_s 放置的副本数量是 IP_s 的 1.5～2 倍以上。出于类似的原因，od_smart 的性能超过了 overlay_s，这是因为 od_smart 使用了 ping 报文来得到客户端和服务器之间的实际 IP 距离。这个观测结果也解释了为什么 od_smart 能得到与 IP_s 相似的性能表现。因此，可以采用一个期

望的缩放因子来“缩放”覆盖的延时，由此来估计真实的 IP 距离，这样就能够降低 ping 探测所引发的流量。

3.7　结论

随着分发系统的日益普遍和全球化，自适应的副本放置和更新分发方法变得越来越重要。本章提出的 SCAN 是一种可衡量、软状态的副本管理框架，它建立在一个利用局部性的分布式对象定位和路由框架（DOLR）之上。SCAN 在需要时生成副本，并将副本自组织到一个应用级组播树中，同时又兼顾到对客户端服务质量（QoS）和服务器容量的约束。本章对 SCAN 进行了基于事件驱动的仿真，结果表明 SCAN 放置的副本数量接近最优，与此同时，SCAN 也提供了良好的负载分配和低延时性能，与基于 IP 组播的静态副本放置相比，带宽消耗也相对较低。不仅如此，在复制和更新成本方面，SCAN 的性能要优于现有的基于 DNS 重定向的 CDN。对于构建全球性对等网络设施而言，SCAN 将很有希望成为其一个关键性的组成部分。

致谢

本章的若干材料出现于 Pervasive’02（the first International Conference on Pervasive Computing）[9]。作者感谢其他合作者对这项工作的贡献：UC Berkeley 的 Randy H. Katz 教授和 John D. Kubiatowicz 教授、UT Austin 的 Lili Qiu 教授。

参考文献

[1] Barbir, A., Cain, B., Douglis, F., Green, M., Hofmann, M., Nair, R., Potter, D., and Spatscheck, O. Known CN request-routing mechanisms. http://www.ietf.org/internet-drafts/draft-ietf-cdi-known-request-routing-00.txt.

[2] BBNPlanet. telnet://ner-routes.bbnplanet.net.

[3] Bestavros, A. Demand-based document dissemination to reduce traffic and balance load in distributed information systems. In *Proceedings of the IEEE Symposium on Parallel and Distributed Processing* (1995).

[4] Bestavros, A., and Cunha, C. Server-initiated document dissemination for the WWW. In *IEEE Data Engineering Bulletin* (Sep. 1996).

[5] Breslau, L., Estrin, D., Fall, K., Floyd, S., Heidemann, J., Helmy, A., Huang, P., McCanne, S., Varadhan, K., Xu, Y., and Yu, H. Advances in network simulation. *IEEE Computer 33*, 5 (May 2000), 59–67.

[6] Castro, M., and Liskov, B. Proactive recovery in a byzantine-fault-tolerant system. In *Proceedings of USENIX Symposium on OSDI* (2000).

[7] Chawathe, Y., McCanne, S., and Brewer, E. RMX: Reliable multicast for heterogeneous networks. In *Proceedings of IEEE INFOCOM* (2000).

[8] Chen, Y., Bargteil, A., Bindel, D., Katz, R. H., and Kubiatowicz, J. Quantifying network denial of service: A location service case study. In *Proceeding of Third International Conference on Information and Communications Security (ICICS)* (2001).

[9] Chen, Y., Katz, R. H., and Kubiatowicz, J. D. SCAN: a dynamic scalable and efficient content distribution network. In *Proceedings of the First International Conference on Pervasive Computing* (Aug. 2002).

[10] Chen, Y., Qiu, L., Chen, W., Nguyen, L., and Katz, R. H. Clustering Web content for efficient replication. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP)* (2002).
[11] Chen, Y., Qiu, L., Chen, W., Nguyen, L., and Katz, R. H. Efficient and adaptive Web replication using content clustering. *IEEE Journal on Selected Areas in Communications (J-SAC), Special Issue on Internet and WWW Measurement, Mapping, and Modeling 21*, 6 (2003), 979–994.
[12] Chu, Y., Rao, S., and Zhang, H. A case for end system multicast. In *Proceedings of ACM SIGMETRICS* (June 2000).
[13] Czerwinski, S., Zhao, B., Hodes, T., Joseph, A., and Katz, R. An architecture for a secure service discovery service. In *Proceedings of ACM/IEEE MobiCom Conference* (1999).
[14] Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. Globally distributed content delivery. *IEEE Internet Computing* (September/October 2002), 50–58.
[15] Fan, L., Cao, P., Almeida, J., and Broder, A. Summary cache: A scalable wide-area Web cache sharing protocol. In *Proceedings of ACM SIGCOMM Conference* (1998).
[16] Francis, P. Yoid: Your own Internet distribution. Technical report, ACIRI, http://www.aciri.org/yoid, April, 2000.
[17] Gifford, D. K., Johnson, K. L., Kaashoek, M. F., and O'Toole, J. W. Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of USENIX Symposium on OSDI* (2000).
[18] Gray, J., Helland, P., O'Neil, P., and Shasha, D. The dangers of replication and a solution. In *Proceedings of ACM SIGMOD Conference* (June 1996), *25*, 2, pp. 173–182.
[19] Guttman, E., Perkins, C., Veizades, J., and Day, M. Service Location Protocol, Version 2. IETF Internet Draft, November 1998. RFC 2165.
[20] Gwertzman, J., and Seltzer, M. World-Wide Web Cache consistency. In *Proceedings of the 1996 USENIX Technical Conference* (1996).
[21] Gwertzman, J., and Seltzer, M. An analysis of geographical push-caching. In *Proceedings of International Conference on Distributed Computing Systems* (1997).
[22] Hildrum, K., Kubiatowicz, J., Rao, S., and Zhao, B. Distributed data location in a dynamic network. In *Proceedings of ACM SPAA* (2002).
[23] Howes, T. A. The lightweight directory access Protocol: X.500 Lite. Tech. Rep. 95–8, Center for Information Technology Integration, U. Mich., July 1995.
[24] Jain, K., and Varirani, V. Approximation algorithms for metric facility location and *k*-median problems using the primal-dual schema and lagrangian relaxation. In *Proceedings of IEEE FOCS* (1999).
[25] Jamin, S., Jin, C., Kurc, A., Raz, D., and Shavitt, Y. Constrained mirror placement on the Internet. In *Proceedings of IEEE Infocom* (2001).
[26] Kistler, J., and Satyanarayanan, M. Disconnected operation in the Coda file system. *ACM Transactions on Computer Systems 10*, 1 (Feb. 1992), 3–25.
[27] Krishnamurthy, B., and Wang, J. On network-aware clustering of Web clients. In *Proceedings of SIGCOMM* (2000).
[28] Krishnamurthy, B., Wills, C., and Zhang, Y. On the use and performance of content distribution networks. In *Proceedings of SIGCOMM Internet Measurement Workshop* (2001).
[29] Kubiatowicz, J., et al. Oceanstore: An architecture for global-scale persistent storage. In *Proceeedings of 9th ASPLOS* (2000).
[30] Li, B., Golin, M. J., Italiano, G. F., Deng, X., and Sohraby, K. On the optimal placement of Web proxies in the Internet. In *Proceedings of IEEE INFOCOM* (1999).
[31] Limelight Networks Inc. `http://www.limelightnetworks.com/`.
[32] Luotonen, A., and Altis, K. World-Wide Web proxies. In *Proceedings of the First International Conference on the WWW* (1994).
[33] Mao, Z. M., Cranor, C., Douglis, F., Rabinovich, M., Spatscheck, O., and Wang, J. A precise and efficient evaluation of the proximity between Web clients and their local DNS servers. In *Proceedings of USENIX Technical Conference* (2002).
[34] Michel, S., Nguyen, K., Rosenstein, A., Zhang, L., Floyd, S., and Jacobson, V. Adaptive Web caching: Towards a new caching architecture. In *Proceedings of 3rd International WWW Caching Workshop* (June, 1998).
[35] Mirror Image Internet Inc. `http://www.mirror-image.com`.
[36] MSNBC. http://www.msnbc.com.

[37] NASA kennedy space center server traces. `http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html`.

[38] Pendarakis, D., Shi, S., Verma, D., and Waldvogel, M. ALMI: An application level multicast infrastructure. In *Proceedings of 3rd USENIX Symposium on Internet Technologies* (2001).

[39] Plaxton, C. G., Rajaraman, R., and Richa, A. W. Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the SCP SPAA* (1997).

[40] Qiu, L., Padmanabhan, V. N., and Voelker, G. M. On the placement of Web server replica. In *Proceedings of IEEE INFOCOM* (2001).

[41] Rabinovich, M., and Aggarwal, A. RaDaR: A scalable architecture for a global Web hosting service. In *Proceedings of WWW* (1999).

[42] Radoslavov, P., Govindan, R., and Estrin, D. Topology-informed Internet replica placement. In *Proceedings of the International Workshop on Web Caching and Content Distribution* (2001).

[43] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM* (2001).

[44] Rodriguez, P., and Sibal, S. SPREAD: Scalable platform for reliable and efficient automated distribution. In *Proceedings of WWW* (2000).

[45] Rowstron, A., and Druschel, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of ACM Middleware* (2001).

[46] Rowstron, A., Kermarrec, A.-M., Castro, M., and Druschel, P. SCRIBE: The design of a large-scale event notification infrastructure. In *Proceedings of International Workshop on Networked Group Communication (NGC)* (2001).

[47] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of ACM SIGCOMM* (2001).

[48] Venkataramani, A., Yalagandula, P., Kokku, R., Sharif, S., and Dahlin, M. The potential costs and benefits of long term prefetching for content distribution. In *Proceedings of Web Content Caching and Distribution Workshop 2001* (2001).

[49] Zegura, E., Calvert, K., and Bhattacharjee, S. How to model an Internetwork. In *Proceedings of IEEE INFOCOM* (1996).

[50] Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., and Kubiatowicz, J. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications* (2003).

[51] Zhuang, S. Q., Zhao, B. Y., Joseph, A. D., Katz, R. H., and Kubiatowicz, J. D. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of ACM NOSSDAV* (2001).

第4章　内容分发和管理

Claudia Canali，Valeria Cardellini，Michele Colajanni 和 Riccardo Lancellotti

4.1　引言

在过去的十年中，互联网从一种以盈利为目的的内容分发手段演变为一种重要的通信媒介，用户通过互联网可以得到迫切需要的信息和服务。互联网的上述成功转变，得益于内容提供商（ICP）对内容分发性能的关注。面对高可用性、可扩展性和网络性能的要求，内容提供商常用的办法是利用第三方的服务来提高内容和服务分发的性能。从技术上讲，内容分发网络（CDN）提供商的目标就是：即使在面临汹涌而至的请求时也能确保内容分发具有足够的性能。

CDN 最初的目标是在互联网上分发静态的网络内容和一些有限大小的视频/音频流。第一代 CDN 最初的设计目标，是用于缓解当静态网页内容经过一些拥堵的对等点（peering points）[⊖]时引起的阻塞。不过，内容分发所面临的环境已经与十年前有了很大的不同，在互联网中需要分发的内容已经变得更加复杂了。目前网络上的内容，根据用户的偏好和需要常具有个性化特点，并需要实现动态的生成。传统的用于静态内容的分发技术已经不能满足新的需要；同时，动态内容的分发存在着一些亟待解决的固有挑战和限制。近年来，CDN 开始支持动态内容的分发，这就使内容提供商可以将 CDN 的优势应用于当前的网络应用和服务。

本章讨论的问题，是如何使用 CDN 进行内容（特别是动态产生的内容和个性化内容）的分发。除了当前 Web 系统的主要功能，本章还介绍如何通过将一个典型的多层 Web 系统的功能复制到 CDN 的节点上，来提高内容分发的性能和扩展性。对于每一种解决方案，除了学术界的最新研究成果，本章同时也介绍了使用这些方案的工业级产品。不仅如此，本章还将介绍每一种基于 CDN 的复制方案的优缺点，并指出哪些情况下可以得到最好的收益而哪些条件下会降低性能。

本章的内容是这样安排的：4.2 节中介绍相关的背景资料，给出一个分发动态和个性化内容的 Web 系统的逻辑各层，并对这些层所使用的主要缓存和复制方法进行了分类；4.3 ~4.5 节详细地解释这些逻辑层如何映射到 CDN 的节点上以加速静态和动态内容的分发；4.6 节讨论的是在一个分布式分发网络设施上，与用户配置信息的管理和复制相关的一些问题；在最后的 4.7 节中对本章进行了总结，给出了若干结论以及研究方向。

⊖　所谓对等点，是指互联网中充当接入点或各子网之间路由交换中心的节点。——译者注

4.2　Web 内容分发系统

本节对那些产生和分发 Web 内容的系统的体系结构进行简要的介绍，并提供一些背景资料。首先，在 4.2.1 节中对多层 Web 系统的各逻辑层进行回顾；其次，在 4.2.2 节中介绍了 CDN 的架构和主要的功能。为了加速动态内容和个性化内容的创建和分发，CDN 在一个 Web 系统的各逻辑层使用了不同的缓存和复制方法。在最后的 4.2.3 节中将对这些方法进行分类。

4.2.1　Web 系统的逻辑层

当前绝大多数 Web 系统是基于一个多层的逻辑架构，这个架构包括 HTTP 接口、应用（业务）逻辑、数据仓库以及可能的与用户有关的身份验证和内容个性化信息，这些层常被称为前端（front - end）层、应用（application）层、后端（back - end）层和用户配置（user profile）层[8]。图 4.1 显示了一个提供 Web 服务的典型系统的结构，从中可以看到组成这个系统的各个逻辑部分以及各层之间最基本的交互。

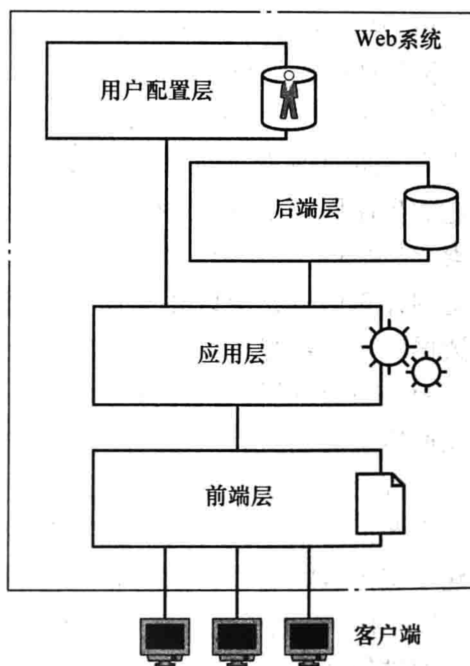


图 4.1　一个 Web 系统的逻辑层

前端层是 Web 服务的接口。该层接受来自客户端的 HTTP 连接请求，进而将来自文件系统的静态内容提供给用户，因此它代表了一个指向中间层应用逻辑的接口。静态内容的分发是一个直接的操作。静态 Web 内容一般存储在一个文件系统中，HTTP 服务器管理客户端对这类内容的请求，从文件系统中取出相应的资料发送回客户端。

前端层处理的静态内容有以下例子：

（1）嵌在一个网页中的网络对象

典型的嵌入对象包括图片、样式表及活动组件，如 flash 动画、Java 程序、ActiveX 控件。

（2）多媒体内容

音频流和视频流是前端层处理的静态内容。为了能够在客户端实现多媒体内容的平滑播放，通常采用基于 HTTP 流的方法，即将多媒体资源分为若干个数据块，然后逐个顺序地发送到客户端。只要收到第一个数据块，客户端就能够开始播放，而不需要一直等到全部资源都接收完毕[26]。

（3）页片（Page fragment）

页片是一个具有独特主题或功能的网页的组成部分[14]。页的每一个片段都被视为一个独立的信息实体。例如，一个门户网站的页面一般包含新闻、特写、链接条和

广告等片段。因为有些片段可能被多个页共用，所以在静态内容的管理中使用片段是为了提高 Web 内容的复用能力。不过，在使用基于页片的静态内容管理时，前端层还要负责在向用户发送网页数据之前将页片组装为网页。

应用层位于 Web 服务的核心：它处理所有的业务逻辑，并计算那些用于生成动态内容的相关信息。在产生所需内容时，应用层常常需要与后端层和用户配置层发生交互，这就要求能够实现应用逻辑与后端层中存储的数据之间的交互，而且也必须能够在需要产生个性化内容时访问用户的配置信息。动态内容是在响应客户端请求时创建的，应用层所创建的动态内容的例子如下：

1）从一个有组织的信息源检索出的答案，如购物车中的商品信息或在一个电子商务网站的搜索结果。

2）动态创建的与数据表示相分离的网络内容，如内容管理系统㊀或基于 XML 的技术[47]提供了实现 Web 文档中结构与内容相分离的机制。在这些系统中，内容（即使它的生命周期相对长一些）从一个模板实时、动态地创建。由于需要进行数据库检索、（可选的）信息处理以及生成 HTTP 输出信息等操作，动态数据的创建会面临一定的计算压力，所以相比而言通过软件（尤其是专门为处理数据而设计的软件，如数据库管理系统）方便地处理数据就显得尤为重要。

3）由用户的社会行为产生的 Web 内容，如论坛和博客的网页为互联网用户提供了交换信息的场所。

后端层为一个 Web 服务管理其主要的信息库。一个典型的后端层由关键信息库和一个数据库服务器组成，其中关键信息库用于产生动态内容。如果再看一下应用层创建动态内容的例子，就能唯一地确定下面的数据库：

1）在一个电子商务站点，使用一个数据库来存储产品的目录，并按产品分类对数据库进行搜索。进而，通过保存有购物车状态或购买产品清单的数据库，就可以管理用户与网页的交互信息。

2）如果是一个内容管理系统，那么在产生网络资源的过程中，创建动态内容的模块对数据库进行访问，从中检索出网页的模板和相应的实际内容。

3）对于论坛或博客等网站，主题、评论内容和邮件一般都存储于一个数据库中。

用户配置层保存用户的偏好和运行环境[16]。这些信息在创建动态内容时被访问，可以提供个性化的信息。存储在用户配置资料中的信息可能源于以下几个方面：

1）由用户提供的信息。一般借助一张信息输入表来增加和修改用户的偏好信息。这种资料的交流可能发生在用户注册时，也可能发生在注册后用户对信息的增添或修改时。

2）通过分析推断出的信息。这种信息是采用数据挖掘等典型方法从网络日志中对用户的行为进行分析后推断出的[10, 21, 27]。一个典型的依靠数据挖掘收集信息的网络服务的例子，是电子商务中使用的产品购买推荐系统[27]，以及根据用户的偏好生成量身定制的广告。

㊀ 若想详细了解目前常用的内容管理系统，可以参见 http：//www.cmsmatrix.org。——原书注

4.2.2　一个简化的 CDN 架构

CDN 架构的设计目标，是通过复制系统资源（即 Web 服务器）的方式来获得高性能和高扩展性，以确保在海量内容下仍能提供高性能的服务。系统资源的复制可以在本地和地理两个尺度上进行。如果是本地复制，则响应用户请求的各台服务器紧密地连接在同一个局域网内。这些服务器一般是共享同一条上行连接，这条上行连接将本地系统连接到互联网。这样的本地系统一般称为集群或簇（cluster）。经过系统资源的复制，若干台服务器组织成一个集群后可以提供更大的计算能力，它们之间能够以快速有效的方式彼此交互[11]。不仅如此，复制还可以提高容错能力，因为一个出错的节点能够很容易地被旁路。

基于局域网的本地系统有很多优点。不过，当网站访问量很大时，本地系统的可扩展性在有效地创建和分发资源方面面临着一些问题。影响本地复制的第一个问题是所谓的“第一公里”（first mile）问题。第一公里是指服务器集群和互联网之间的网络连接，它是影响端到端性能的系统瓶颈。不仅如此，它还是一个潜在的失效单点。无论集群的计算能力如何，集群内通信不畅、外部某个路由器的失效和拒绝服务（DoS）攻击都可能会造成集群无法提供服务。当需要更好的性能和可扩展性时，在地理这个尺度上复制网络设施中的一些资源会有所帮助。

图 4.2 显示的是一个 CDN 在地理上的分布架构简图。在一个典型的 CDN 中，服务器分为两类：边缘服务器（edge server）和核心服务器（core server）[18, 36]。边缘服务器是源服务器（核心服务器）在“互联网边缘”的副本，主要负责与使用网络服务的客户端实现交互。边缘服务器是指那些与客户端距离最近的服务器，一个典型的例子是多 ISP 的网络接入点（Points Of Presence，POP）。通过基于 DNS 的重定向方法，客户端的请求从源服务器转发到边缘服务器[12, 36]。在这个方法中，DNS 服务器需事先做某些调整，以便当其收到对主机名的查询时，可以将其解析为合适的边缘服务器的 IP 地址。寻找最佳边缘服务器的算法一般较为复杂，需要考虑地理意义上的距离、网络意义上的距离、网络连接情况和边缘服务器的状态等因素。

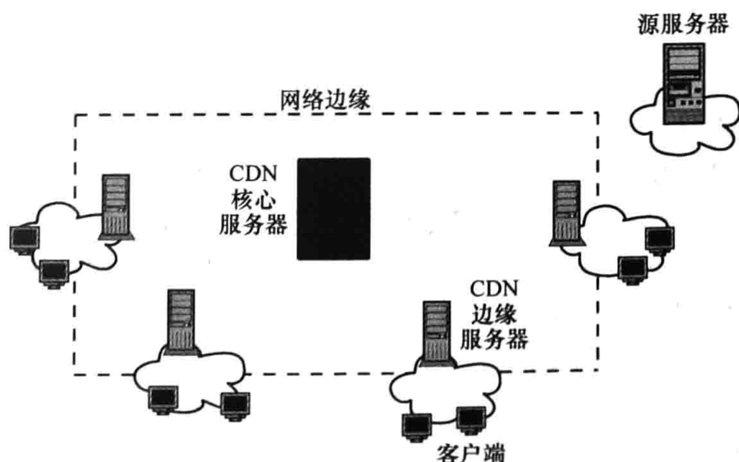


图 4.2　一种简化的 CDN 架构

核心服务器是一个逻辑实体，负责处理网络设施的管理、服务请求分配的协调以及维护请求列表。核心服务器可以是一台单独的高性能服务器，也可以是一个多集群（multi－cluster），即若干个服务器集群的集合。多集群中的各个集群之间相互协作，对外表现为一台虚拟的高可用性和高计算能力的计算机。

4.2.3 内容的创建和分发的加速

互联网演化的趋势是对内容分发的可扩展性和高性能提出了越来越大的需求，这使内容提供商开始转而依靠 CDN。同时，CDN 也需要为内容提供商研究加速内容分发的技术。

为了理解 CDN 如何加速互联网内容和应用的分发，可以把关注重点放在源服务器和边缘服务器，因为它们是网络设施中涉及内容分发过程的主要部分。解决可扩展性和性能问题的两个经典思路是缓存和复制。CDN 将源服务器上的一些逻辑层复制到边缘服务器上。如前面所述，互联网系统中有 4 个逻辑层次，因此可以考虑以下 4 种方法，示意图见图 4.3。

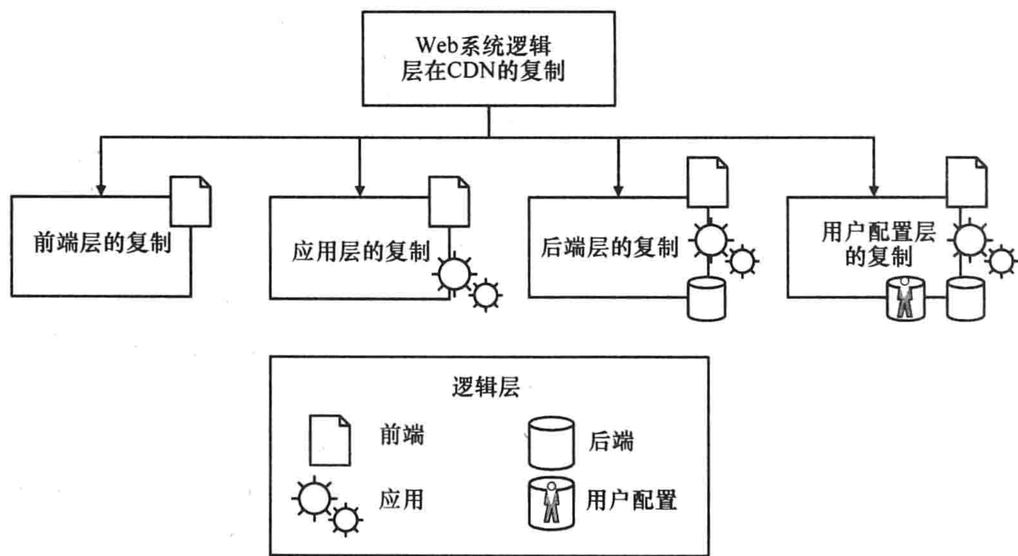


图 4.3　CDN 设施中 Web 逻辑层的映射

1. 前端层的复制

这是第一代 CDN 使用的典型方法。在该方法中，边缘服务器（当时称为代理缓存服务器，surrogate server）只负责静态内容的管理，其作用类似反向代理服务器（reverse proxy），实现内容分发的加速（内容的存储可以在文件系统级实现）[36, 49]。被复制的网络内容可以是整个网络对象（如内嵌对象或多媒体资源），也可以是 Web 片段（这时，复制可以使用更精细的方法）[14]。

2. 应用层的复制

这种方法也称边缘计算[44]，是把网络应用程序或组件直接转移到边缘服务器上[18, 37]，目的是在接近客户端的地方创建动态内容。因此，CDN 可以提高动态内容

的分发性能。

3. 后端层的复制

边缘服务器不仅实现动态内容的创建，而且还可以管理创建时使用的数据。源服务器只负责网络设施的管理，并保存数据的一份主副本。

4. 用户配置层的复制

边缘服务器还要管理用于产生个性化内容的数据库[42]。

4.3 前端层的复制

前端层的复制是为了提高静态内容分发时的性能和可扩展性。如图 4.4 所示，这些内容缓存在 CDN 边缘服务器中。将静态内容的分发工作转由边缘服务器来完成，是为了解决可扩展性的问题，因为这样做可以避免出现在对等点和广域网链路处的网络拥堵风险，而这两个地方的拥堵是网络延时的主要原因[36]。

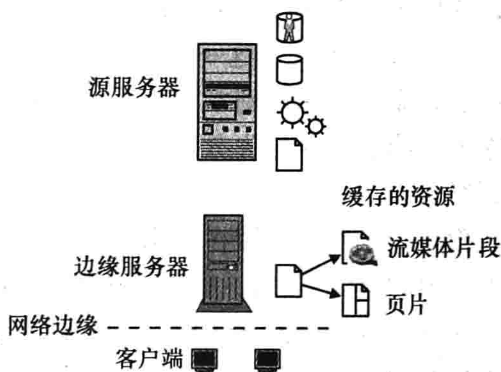


图 4.4　前端层向边缘服务器的复制

通过使用一个第三方的网络设施来加速静态内容的分发，是一种提高内容分发性能的常用方法。这种做法可以追溯到第一代 CDN，如 Adero CDN 或 Akamai 的媒体分发服务[2]。不过，静态内容的分发仍然是一个棘手的工作，因为富媒体（rich - media）⊖内容的使用已经变得越来越多，它产生的流量正在成为互联网流量的一个重要组成部分。将这类媒体内容的分发转移到接近客户端的地方，可以获得很大的好处，原因有二：首先，由于这类内容的尺寸庞大，在对等点处的网络延时会使用户明显感觉到性能的下降；其次，借助常用的 HTTP 流技术可以降低分发时间上的不确定性，使媒体的播放变得更加流畅[26]。

由于多媒体内容的尺寸较大，所以通常只把每个多媒体内容中用户最感兴趣的部分（而不是全部）缓存在边缘服务器中[15, 25]，称分段缓存（segment caching），如图 4.4 所示。一个多媒体资源中每个片段（fragment）的受欢迎度由用户的访问模式决定：如果用户采用（从头）连续访问的模式，那么通常使用连续缓存的方法，也就是存储资源的起始部分，以减小缓冲时间；如果用户的访问模式主要是媒体中的随机定位，那么就要使用不同的缓存技术，如交错缓存方法，这样可能会更加有效[25]。

当待分发的 Web 内容是由片段（fragment）组装而成时（见图 4.4 中被缓存的资源），有人提出了将流内容分割为段（segment）的方法。这个方法要求边缘服务器做

⊖ 区别于单一的图片、视频等传统媒体，rich - media 是具有复杂视觉效果和交互功能的媒体，可以由网页、图片、超链接、在线视频的即时播放等资源组合而成。rich - media 丰富了网络媒体的内容，呈现出更好的效果，主要用于网络在线广告等。国内一般译为“富媒体”。——译者注

更多的工作，其前端层必须包含两个功能：为每个片段提供单独的缓存和片段的组装。作为一个独立的信息实体，每个片段可以包含描述其缓存能力的配置信息，该信息用来指示片段能否被缓存和它在网络上的存活时间（Time - To - Live，TTL），这就可以在片段（而不是在网页）的粒度上管理内容的新鲜度㊀和生命周期。

当一个用户请求网页时，边缘服务器在它的缓存中找到该网页的片段，然后立刻将其组装起来。只有那些失去时效或没有被缓存的片段才从源服务器取。因此，在边缘服务器上使用基于片段的缓存和动态组装方法，对源服务器而言有两个好处：一是它不必完成组装页面的任务；二是它一般只需要分发页面的一小部分内容，也就是边缘服务器上那些失效或缓存中没有的片段。对于大部分由动态创建的网页构成的网络资源，基于片段的缓存技术可以在距离终端用户最近的互联网边缘向用户提供这些内容，从用户对性能的感受可以发现，这一技术已被证明能够有效地改善响应时间[38, 51]。不仅如此，这一技术同样对边缘服务器有益：那些被不同网页共享的同一个片段只需保存一份副本，这就大大提高了磁盘空间的利用率；同时，它也能够降低边缘服务器上缓存的失效频度，因为网页中的不同部分只有在过期的时候才标记为“失效”。

衡量基于片段的缓存技术的常用标准是ESI（Edge Side Includes）[19]，ESI是一种基于XML的标记语言，能够使用XML来区分内容是否应该缓存。内容提供商根据其开发环境制订相应的ESI规范，用来设计和开发业务逻辑以实现网页的产生和装配。除了将页面分解为片段这一基本功能（即使在一个有条件的方式下），ESI的功能还包括片段不可用时的异常处理，以及对被缓存片段的失效性做出判断，这就使ESI可以提供比TTL机制更强的一致性确保机制[19][29]㊁。

基于片段的网页发布和缓存技术已经被很多公司和商业产品使用，如Akamai公司基于ESI规范的EdgeSuite网络[2]，以及IBM公司的WebSphere Edge Server[28]。Challenger等人提出了一个基于片段方法和兼容ESI的Web发布系统的大规模部署方案[14]，这个系统能够用片段构造复杂的对象，安装在IBM托管的网站上用于处理主要的体育事件。在参考文献[14]中，也讨论了对受片段变化影响的网页进行检测和更新的问题。该文献提出的方法，是根据一致性要求采取基于图搜索的不同算法。Yuan等人研究了边缘服务器上的4种不同的缓存和负载分流（offloading）策略[51]，并使用了一个有代表性的电子商务基准进行了评估。评估结果显示，一个简单的将网页分解功能分流的策略，可以非常有效地降低服务器的负载和延时。

大多数支持基于片段的缓存技术的边缘服务器并不提供缓存之间的任何协作，也就是说，这些缓存完全被作为独立的实体。这就使我们无法享受协作带来的好处，而采取协作的方式可以使边缘服务器发挥其潜在的缓存能力。Akamai的EdgeSuite在这个方向上进行了一些努力，但仅限于边缘服务器之间的协作上。最近，Ramaswamy等人[39]在大范围网络中边缘服务器的协作上解决了一些重要问题，他们提出了基于动态

㊀ 新鲜度（freshness）用于指示数据在经过网络传输后是否保持了正确性。——译者注

㊁ 对缓存一致性机制的分析可参见第5章。——原书注

哈希的文档检索/更新协议的低成本协作技术，也研究了如何实现单个边缘服务器的失效恢复。

基于片段的方法也面临着一些问题的困扰，包括与动态内容的类型相关的应用性问题以及网页的片段分解问题，而该方法的主要缺点都与这些问题有关。的确，如果用户的请求序列具有高度的局部性，同时源服务器上的内容更新不频繁，基于片段的缓存就能有效地应用。这个条件保证了片段的可缓存性参数足以用来管理内容的新鲜度，从而减轻了源服务器给缓存中的片段设置“失效”标志的负担。不仅如此，因为缓存、划分片段、组装片段等工作都是在每一个具体应用上进行，所以该技术很缺乏透明性。例如，由于ESI代码必须加到原始代码上，这就需要完全修改网页的代码，因此ESI的性能就取决于网页的结构。对于同时为多个内容提供商分发内容的边缘服务器来说，上述对片段的人工辨识和标记几乎是无法承受的。为了避免采用人工方式将网页划分为片段，Ramaswamy 等[38]基于对动态网页的详细分析（根据其共享行为和变化模式），提出了一个自动检测网页中片段的方法。

4.4　应用层的复制

如果在CDN中只复制前端层，那么在根据互联网应用逻辑产生动态内容的源服务器上，应用层就成为一个性能上的瓶颈。应用层复制（一般称为边缘计算[18, 45]）的目标是改善动态内容的分发，采取的方式就是将应用层的工作从源服务器上转移出去。互联网应用的代码被复制到多个边缘服务器上，但数据仍然是集中管理的，只是将计算工作推到网络边缘，如图4.5所示。

在边缘计算中，每个边缘服务器有一份应用代码的完整副本，而后端层仍然在源服务器中，也就是说，所有的边缘服务器继续共享一个中央数据库。相对于应用层和数据层采取集中管理而只复制前端层的方法，将计算任务推到网络边缘可以降低源服务器的工作负担，使CDN可以获得更好的效率、性能和更高的可用性。

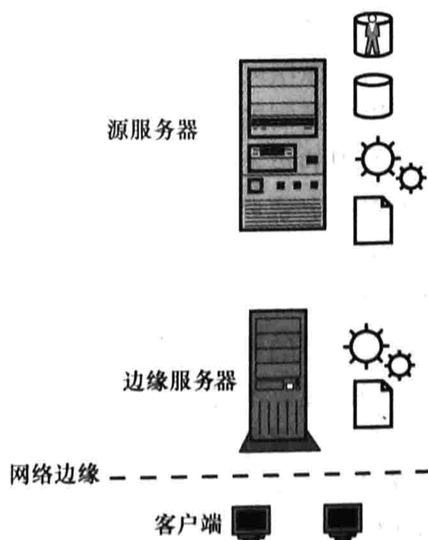


图4.5　应用层向边缘服务器的复制

根据边缘服务器区分事务性请求和非事务性请求的能力，可以确定两种不同架构的解决方案。一个事务性的请求是一个数据库操作的原子集合，这类请求通常是给数据库的某部分加锁并修改数据库中的记录；而非事务性的请求对数据的操作方式是只读的。如果边缘服务器不能区分请求的类型，则边缘服务器上的Web服务器将所有的请求直接转给其本地应用层处理，随后本地应用逻辑访问位于CDN核心上的中央数据层。如果边缘服

务器能够区分请求的类型，则它只把非事务性的请求转到本地应用层，而将事务性请求直接转发给源服务器上的应用层，在那里处理事务和访问集中管理的数据库。

在应用层复制的方法中，一般由 CDN 核心充当协调任务，负责把应用迁移和（或）复制到边缘服务器上，并保持对副本的跟踪。有时 CDN 核心也需要负责维护这些副本与原件的一致性，为了实现这个功能，它采取一个简单的基于服务器的失效性检查，也就是说，当原件被开发人员修改后负责更新边缘服务器上的副本。

边缘计算已经被应用在很多商业产品和学术项目中。例如，它是 Akamai 公司开发的产品 EdgeComputing 的核心[3]，该产品托管那些由用户提供安装到网络边缘端的应用服务器上的 J2EE 组件。EdgeComputing 使用一个两级模型来复制应用层：包含着表示逻辑的 JSPs 和 servlets 被部署在 Akamai 网络的边缘服务器，而业务级组件由于与后端层或数据库联系紧密，所以仍放在源服务器的 CDN 核上。

通过多年的研究，进程的迁移问题已得到了解决。一个应用在复杂系统上的例子是 vMatrix[9]，它将应用的整个动态状态从一台服务器迁移到另一台服务器㊀。不过，Web 应用并不需要在任意时刻进行实际的应用迁移，而是可以在需要的时候进行[37]。因此，可用于 CDN 环境下的一个明显的简化方法是在边缘服务器进行应用的自动部署。Rabinovich 等人提到的 ACDN（Application CDN）[37]是一个采用这一自动部署概念的应用分布框架，根据需要在统一的协调下进行对应用的动态复制。这个框架的实现是基于一个元文件（meta - file），该文件包含构成应用的文件列表和一个初始化脚本。

由 Zhao 和 Schulzrinne[52,53]提出的 DotSlash 框架是另一个学术项目，它使用一个动态脚本复制技术来管理动态内容。DotSlash 并不是为大规模 CDN 设计的，它动态地提供救援服务器作为缓存代理服务器，以此来提供一个系统以解决对网站性能造成影响的负载瞬间激增问题。

应用层复制的特点是能够实现那些具体和特定应用的定制化，应用层复制的方法既不是通用的也不是透明的。它需要为每个具体的应用实现定制化，因为它需要通过人工配置决定哪些组件要被转移以及在哪里部署应用。例如，在前面提到的 ACDN 中，应用可以动态地部署及重新部署，但仍然需要人工的干预，如为每个需要复制的应用创建元文件。应用的定制化会大大增加其管理者的成本，而且编写过程中也容易引入拼写之类的错误。参考文献[33]在自动判断如何进行互联网应用的复制方面做出了一些研究。不过，这些研究主要是着眼于小规模的情况，可能不适用于那些有成千上万个边缘服务器的大规模 CDN。

应用层复制方法面临的其他不足之处与源服务器对数据的集中管理有关。这种对数据进行集中管理的架构有两个缺点：首先，在一个大规模 CDN 中，边缘服务器之间距离很远，每次数据访问都会发生广域网延时；其次，中央数据库可能很快就会成

㊀ 在该方法中，借助一个 VMM（Virtual Machine Monitor）的概念，将计算机 A 的状态（CPU、操作系统、代码、库等信息）封装到一个 VM（Virtual Machine）文件中，凭借 VM 文件就能够在同样运行 VMM 软件的计算机 B 上，将计算机 A 实例化和模拟。这样就解决了代码对其编译环境的依赖问题。——译者注

为一个性能上的瓶颈，因为它需要响应整个系统中对数据库的所有请求。因此，应用层复制方法只适用于那些对中央数据库访问极少的网站。

高度集中的数据层管理方式限制了整个CDN的可扩展性，下一节将讨论其他一些方法，这些方法旨在缓解数据集中管理导致的瓶颈问题。因此，进一步的措施是在数据层使用缓存和复制技术，从而将源服务器的功能转移到边缘服务器。

4.5 后端层的复制

因为在一些互联网应用中，瓶颈位于后端层[13]而非应用层，所以边缘计算方法可能无法解决每一个可扩展性问题。在这种情况下，为了解决可扩展性问题，可以将应用数据交给一个第三方（CDN）来管理，由CDN代替源服务器的后端层对边缘服务器应用层发出的数据查询做出响应。

很多学者从数据库的角度对数据复制方案进行了广泛的研究[23]。在本节中，分析的范围仅限于对Web系统后端层数据的复制。参考文献[43]对这个范围内的各种方法进行了总结：对于存储在后端层的数据，其复制可以是完整的也可以是部分的，如图4.6所示。数据的部分复制可以通过两种方式实现：一是将大部分常见的对数据库操作（query）的返回结果进行缓存（内容未知的缓存）；二是主动地复制后端层的部分数据，选择复制哪些数据取决于使用模式、网络和基础设施的状态[44]（内容已知的缓存）。

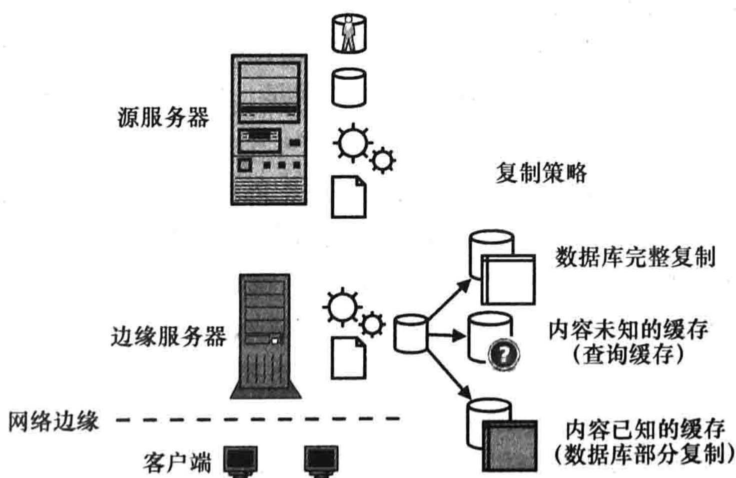


图4.6 后端层向边缘服务器的复制

由于互联网应用对数据库存在不同的访问模式，所以不能简单地认定哪些方法最优。Gao等人在参考文献[22]中提出的不同复制策略，取决于不同Web应用的特性。

4.5.1 内容未知的缓存

在这种缓存策略下，边缘服务器中被缓存的数据是前一次对数据库操作的返回结果。这样，当再次收到相同或类似的请求时，服务器就可以在本地对其处理，这不仅

改善了性能，也能减轻源服务器后端层的负担。

这种通过缓存数据库操作结果来实现后端层复制的方法使用得非常广泛，如GlobeCGC[40]系统。近来，有学者设想通过该缓存机制实现后端层的动态复制，从而可以改善可扩展性。例如，当出现预期之外的流量激增时，DotSlash框架的QCache模块[53]使用一个网站合作协议临时启用一个分布式的缓存设备以减轻过载状况。

对数据库操作结果进行缓存的效果取决于缓存命中率。为了提高命中率，可能需要放宽缓存机制中对内容特性的要求。为此，可以使用稍微复杂一些的匹配引擎：在处理一个新的数据库操作时使用缓存中已有的结果进行响应，而不是由源服务器处理。DBproxy[4]的一个独特之处就是提供了对这个匹配引擎的支持。参考文献[30]中提出了一个有效合并缓存数据的方法，即每次数据库操作的结果被加入到后端数据库的一个（初始为空的）副本中。DBCache[10]在表的层面上支持数据库的缓存，允许在边缘服务器上缓存中央数据库服务器中那些表的整个内容或一个子集。

缓存机制应该确保缓存数据的一致性。既然传统的基于TTL的方法无法对每个Web应用都适用，因此出现了一些与特定应用相关的一致性增强机制。例如，Olston等人使用一个多播消息系统将数据失效的消息发送到每个缓存[32]，而Tolia等人[48]使用哈希函数确保不会使用失效的数据。

在边缘服务器上实现对数据库操作结果的缓存是一个重要的特性，目前被用于很多的商业产品，包括BEA WebLogic和IBM Web - Sphere。特别是IBM Web - Sphere，它通过一个物化查询表（Materialized Query Table，MQT）实现对数据库操作结果的缓存。MQT是一张表，存储预先计算的对一个或多个数据库表的操作结果。在创建和填写MQT之后，对于随后收到的一个任意的数据库请求，如果MQT对它实现了全部或部分的匹配，那么这个数据库请求就得到了满足。BEA WebLogic借助EJB提供了一个类似的功能。

即使采取了一些一致性增强机制，网络设施中的网络层也可能会导致边缘服务器上的数据相对于中央数据层的当前状态而言是失效的。这对于以读为主的应用场合（在这里，网络应用几乎不需要执行事务性操作）也许不是个问题。但是，对于一类重要的应用（如涉及支付操作），事务性操作是必须存在的，数据库的修改也是频繁的。在这种情况下，对数据库操作结果的缓存可能就不再是一个可行的选择。而且，数据库缓存技术只是对那些在数据层反复执行相同操作的应用来说是适用的，而对于那些不具备这种时间局部性的应用，对边缘服务器数据层进行部分或全部复制可能会更有效。

4.5.2 内容已知的缓存

在基于内容已知的缓存技术中，每一个边缘服务器运行自己的数据库服务器，该服务器包含一个中央数据库的部分视图。部分数据复制的典型方法是根据访问模式将数据库的一部分转移到边缘服务器。既然目标是降低对用户请求的响应时间，那么在决定副本放置的算法中（如HotZone算法），通常要在性能建模时考虑网络延时[46]。

复制机制的一个典型的例子是GlobeDB[43]，它在数据分区对数据库进行部分的

复制，以降低因数据库内容修改而产生的流量。但是，这个方案需要一个拥有完整数据库的特定服务器去执行复杂的数据库操作，因此这个服务器可能会成为新的吞吐量瓶颈。GlobeTB[24]对GlobeDB进行了改进，不仅要减小延时，同时还要增加被复制数据库的吞吐量。为此，GlobeTP降低了对单一中央主数据库的要求，避免在源服务器的后端层出现瓶颈风险。

与前述对数据库操作结果的缓存一样，对数据库做部分的复制也可能面临一致性的问题。参考文献[35]引入中间件Ganymed来解决在被复制的后端层面临数据变化（即接收到修改、删除、插入的指令）时如何确保数据一致性的问题，Ganymed把修改操作与只读类事务分开，将前者转到一个中央主数据库服务器处理，而将后者交由只读的数据库副本处理。

很多商业产品也支持数据库的部分复制。例如，mySQL数据库管理系统支持多个副本的数据分区，IBM DB2和Oracle也具有类似的特性。不过，在大多数情况下，数据库部分复制的方案被设计用于对资源（即数据库集群）的一个本地副本进行管理，需要在所有数据库分区之上有一个中央管理器来处理和分发请求和事务。这个方法不能直接应用于一个大范围网络内资源的复制，因为一个中央管理器的存在会影响系统的可扩展性。由于这个原因，大多数商业产品更多地依靠对数据库操作结果的缓存而不是数据库复制策略。

4.5.3 数据库整体复制

服务器整体复制机制负责维护数据的所有副本在每个位置上的一致性。通过将服务器的副本转移到若干个边缘服务器中，并保持数据库的所有这些副本相互一致，那么就有可能在无须修改每一个应用的前提下在网络的边缘完全实现动态内容的分发。不过，对数据库复制的管理也存在严重的一致性维护问题，当客户端的请求使数据被频繁修改时这个问题的解决就显得尤为重要。这是一个众所周知的问题，数据库研究人员长期以来一直在试图解决这个难题。

按照习惯做法，数据复制是通过延时（lazy）或即时（eager）写更新传播来实现[23]。在即时（或称同步）方法中，副本之间的数据同步发生在事务提交之前；而在延时方法中，数据的更新则是在事务提交后才进行传播。即时方法有较好的容错性，而且提供了与单个数据库相同的正确性保证。但是，它在性能和可扩展性方面可能会变得不实用，从而使其面临着很严重的局限性[23]。另外，延时方法有很好的性能和可扩展性，因而在商业产品上受到青睐。但延时方法带来了一些新的问题：因为事务可能会读到失效的数据，且更新事务之间的冲突可能会很晚才被发现，这就需要对潜在的冲突问题进行解决。

在Web环境下，管理数据库复制最简单的方法是基于源服务器中的原件和边缘服务器中的副本。只读业务完全可以由边缘服务器通过访问其数据库副本来完成。不过，对于需要执行更新操作的事务（以写为主的场合），所有对数据库的访问都被重定向到源服务器中数据库的原件上，这个中央数据库再将需要更新的数据定期传播到边缘服务器中的副本。这种方法的一个缺点是，边缘服务器必须清楚应用的含义，因

为它需要知道请求触发的是更新事务还是只读事务。不仅如此，这个解决方案中数据副本间的一致性是通过延时更新传播方法实现的，这带来两个不利影响：首先，边缘服务器中存储的数据可能会失效；其次，一次崩溃可能会导致数据的丢失。

在 Web 环境中，数据库完全复制面临很多挑战。到目前为止，很多数据库复制技术都假设数据库副本之间通过局域网互联。近年来，数据库研究人员提出了很多复制协议，保证数据在局域网中的一致性和较好的性能。对于 Web 环境，本节只提及了一些解决本地分布式 Web 系统下数据库复制的研究工作。有兴趣的读者可以参见参考文献[31]，该文献基于组通信对数据库复制系统进行了全面的分析。Amza 等人[5]提出的延时复制方案，可以通过减少冲突的数量提供序列化能力和吞吐量定级。这一早期工作目前已经有了分布式的版本，能够提供很强的一致性并避免死锁问题[6]。在该学者的最近一篇文献中，研究了如何将数据库操作结果的缓存和集群复制方案进行结合[7]。参考文献[34]使用一个中间件来实现一致和可扩展的数据复制。

在 CDN 中，数据库的副本在地理上分布于一个 WAN 中，如果 Web 应用产生了大量的数据库更新，随之而来的巨大流量将会使 WAN 出现过载并对性能产生负面影响，这是因为每个更新都要被传播到所有其他的副本以保证所有副本数据的一致性。参考文献[31]对那些通过组通信实现 WAN 中数据一致性的数据复制技术进行了性能分析，不过，该文献中对可扩展性的分析只限于 8 个副本。因此，可以得出结论：WAN 中数据库复制的可扩展性和性能大体上仍是一个尚待解决的问题，还需要进一步的研究。

4.6 用户配置层的复制

用户配置层取决于一个实现数据存储的数据库，即后端层。因此，对用户配置信息复制的可能方法已经在 4.5 节描述。但是，这一层的访问模式与后端层显著不同。

特别地，用户通常只与边缘服务器交互，因此在某个用户与网站之间的整个交互期间，用户的配置信息只被一个边缘服务器访问。这种访问模式对一致性和复制策略产生了重大影响。事实上，根据用户的访问模式，全部用户的配置信息可以被分割并分布在各个边缘节点上。既然不需要复制，那么一致性问题就变成只需确保用户配置信息在边缘服务器和源服务器之间的一致性。因此，管理用户配置信息的主要方法也就仅限于对前面提到的内容未知的数据或内容已知的数据的缓存，因为复制整个数据库显然是不必要的。

然而，需要指出的是，即使用户在整个会话期间只访问一个边缘服务器，但由于用户完全有可能在下一个会话时转移到另一个边缘服务器，所以有必要确保用户的配置信息在用户转移到另一个边缘服务器时跟着迁移过去，如图 4.7 所示。在大部分的后端层数据复制策略中，对这种情况的支持并没有得到优化。近年来，一些研究工作试图解决这个配置信息迁移的问题。CONCA[41]是一个通用的数据缓存框架，目的是通过允许数据随用户迁移来支持用户的移动性。这个框架已得到进一步扩展，以实现 Tuxedo 中个人数据的管理[42]。

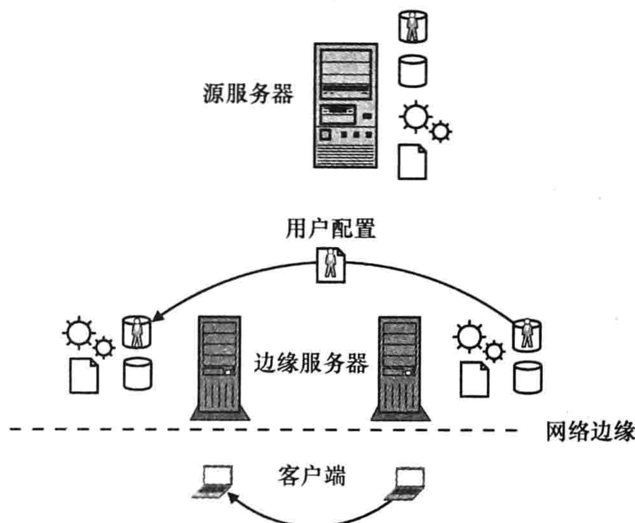


图 4.7　用户配置层向边缘服务器的复制

除了用户相关信息的复制，用户配置层必须完成的关键操作还包括这类信息的创建和更新。目前，用户配置信息的更新或者由用户通过 Web 表单的方式手动更新，或者基于用户的行为方式通过 Web 系统自动更新。用户配置信息的存储和收集方式取决于所采用的 Web 服务。下面将讨论产生个性化内容的一些典型的例子。

1. 通过外部数据源的集合产生个性化内容

这种服务被很多个性化门户网站（如 myYahoo、iGoogle）采用，向用户提供个性化的新闻源，典型的方式是通过 XML - RSS 从异构数据源获取。用户配置信息包含对用户有用的信息以及个性化内容的提交方式。当用户在个性化门户网站注册时，用户通过填写网页上的表单提供与新闻源订阅相关的信息以及他们所喜好的信息显示格式。

2. 协同过滤

这种服务支持用户之间的交互，允许用户对其他用户和话题做出反馈。这种个性化内容产生方式通常用于建议系统，该系统根据对类似用户行为的分析或用户在社交网络中的信誉级别，为购买商品提供建议。用户信息的收集有各种方式，或通过用户输入（例如，在这样的情况下用户的信誉是基于其他用户的反馈），或通过间接的信息采集。例如，按照预先确定的配置信息，通过挖掘用户的购买行为来对用户群进行分类[27]。

3. 位置和基于环境的服务

这种服务根据用户的地理位置来产生个性化内容。通过对数据相关信息的分析或用户访问服务时提供的信息，来确定用户的位置。将用户所处的位置与地理数据进行比较，根据用户的位置和周围环境（可能还结合用户的偏好）进行用户静态和动态内容的产生和分发。

这些例子表明，即使有些信息是由用户提供的，但存储在用户配置信息中的相当一部分数据是通过日志文件的数据挖掘、cookie 和用户单击历史来推断的。以现有的

信息收集技术，可以提取用户的一些令人感兴趣的信息，包括敏感数据，如政治、身体状况和性别特征。此外，大多数的技术对用户来说几乎是透明的，用户在使用网络服务时往往完全没有意识到其存在。近几年出现了一些未经授权的用户信息收集活动，如 doubleclick. com 商业广告服务。一些商业服务（如搜索引擎）均与 doubleclick. com 有关。商业站点使用 cookie 来监视其访客的活动，收集的任何资料都存储在 doubleclick. com 数据库中。然后，这些用户信息被 doubleclick. com 用来选择更适合用户的广告条。

滥用个人信息的现象引起了人们对下述问题的关注：是否应该和如何让用户知道自己的信息正在被收集。由日志数据的挖掘和 cookie 的分析引发的对隐私问题的关注，促进人们研究新的机制来确定什么样的信息可以从用户的行为中提取出来以及应该如何使用。隐私偏好选项平台（Platform for Privacy Preferences，P3P）[17] 是一个旨在解决这一问题的方案：每个遵守 P3P 标准的网站必须提供一个 XML 格式的描述，规定收集哪些数据、数据如何管理、数据应该存储在哪里。完全遵守 P3P 标准会对用户配置信息的自动复制产生一些限制，因为必须确保每个用户配置信息的副本都应该有一个适当的访问级别，这样才能有效地保护用户的隐私。

4.7 结论和尚待解决的问题

静态和动态内容的分发可以由一个第三方（即 CDN）通过复制 Web 系统中的一些层来实现加速。本章对 Web 系统中每个逻辑层的复制进行了分析。对于每一层，本章讨论了内容分发领域的研究成果，同时也说明了 CDN 行业如何使用复制技术提高内容分发的效率。本章的分析表明，当内容提供商想加速静态多媒体内容的分发时，前端层的复制是一个合适的选择；而如果 CDN 用来加速动态内容的分发，则需要进行应用层的复制。复制方法对性能的提高取决于对数据的访问模式，对于某些 Web 应用来说后端层可能是一个瓶颈，这时就必须对后端层进行复制。

在内容分发的研究中存在着若干有待解决的问题。事实上，即使一些加速动态内容分发的方法已经提出并被业界接纳，这些办法的有效性在很大程度上也仍极大地取决于应用的访问模式。特别地，存在于后端层的瓶颈风险仍是阻碍动态 Web 内容分发可扩展性的一个主要问题。由于 Web 内容和应用的发展，这个问题即使在未来几年也很可能仍是一个主要的研究课题。Web 2.0 正推动 Web 向两个主要趋势发展：越来越多的个性化内容和产生大量上传流量的 Web 新使用模式。个性化的（和不可缓存的）内容和内容的高更新率降低了缓存机制的效率；而数据一致性协议的使用导致了维护成本的增长。此外，用户个人信息的存在，引出了用户配置信息中哪些允许复制、哪些不允许复制的边界问题，因为内容提供商必须保护用户敏感信息的私密性。随着 Web 2.0 的发展和用户移动性的普遍，用户在边缘节点之间移动，访问的局部性将被打破，这就使情况的复杂性进一步增加。对于 CDN 运营商和内容分发策略的研究者来说，应对这一发展变化将是他们面临的下一个挑战。

参 考 文 献

[1] Agostini, A., Bettini, C., Riboni, D.: Loosely coupling ontological reasoning with an efficient middleware for context-awareness. In: Proc. of Mobiquitous 2005. San Diego, CA (2005)
[2] Akamai: (2007). http://www.akamai.com/
[3] Akamai EdgeComputing: (2007). http://www.akamai.com/html/technology/edgecomputing.html
[4] Amiri, K., Park, S., Tewari, R., Padmanabhan, S.: DBProxy: A dynamic data cache for Web applications. In: Proc. of 19th IEEE Int'l Conf. on Data Engineering, pp. 821–831. Bangalore, India (2003)
[5] Amza, C., Cox, A., Zwaenepoel, W.: Conflict-aware scheduling for dynamic content applications. In: Proc. of 4th USENIX Symp. on Internet Technologies and Systems (2003)
[6] Amza, C., Cox, A., Zwaenepoel, W.: Distributed versioning: Consistent replication for scaling back-end databases of dynamic content web sites. In: Proc. of ACM/IFIP/Usenix Middleware Conf. (2003)
[7] Amza, C., Cox, A., Zwaenepoel, W.: A comparative evaluation of transparent scaling techniques for dynamic content servers. In: Proc. of IEEE Int'l Conf. on Data Engineering (2005)
[8] Andreolini, M., Colajanni, M., Mazzoni, F., Lancellotti, R.: Fine grain performance evaluation of e-commerce sites. ACM Performance Evaluation Review **32**(3) (2004)
[9] Awadallah, A., Rosenblum, M.: The vMatrix: A network of virtual machine monitors for dynamic content distribution. In: Proc. of 7th Int'l Workshop on Web Content Caching and Distribution (2002)
[10] Bornhovd, C., Altinel, M., Mohan, C., Pirahesh, H., Reinwald, B.: Adaptive database caching with DBCache. IEEE Data Engineering Bulletin **27**(2), 11–18 (2004)
[11] Cardellini, V., Casalicchio, E., Colajanni, M., Yu, P.S.: The state of the art in locally distributed web-server systems. ACM Computing Surveys **34**(2) (2002)
[12] Cardellini, V., Colajanni, M., Yu, P.: Request redirection algorithms for distributed web systems. IEEE Tran. on Parallel and Distributed Systems **14**(5) (2003)
[13] Cecchet, E., Chanda, A., Elnikety, S., Marguerite, J., Zwaenepoel, W.: Performance comparison of middleware architectures for generating dynamic Web content. In: Proc. of 4th ACM/IFIP/USENIX Middleware (2003)
[14] Challenger, J., Dantzig, P., Iyengar, A., Witting, K.: A fragment-based approach for efficiently creating dynamic Web content. ACM Transactions on Internet Technology **5**(2), 359–389 (2005)
[15] Chen, S., Shen, B., Wee, S., Zhang, X.: Adaptive and lazy segmentation based proxy caching for streaming media. In: Proc. of ACM NOSSDAV (2003)
[16] Colajanni, M., Lancellotti, R., Yu, P.S.: Scalable architectures and services for ubiquitous web access. In: Tutorial notes in 2006 World Wide Web Conf. (2006)
[17] Cranor, L.: Web Privacy with P3P. O'Reilly (2002)
[18] Davis, A., Parikh, J., Weihl, B.: EdgeComputing: Extending enterprise applications to the edge of the Internet. In: Proc. of 2004 World Wide Web Conf. (2004)
[19] Edge Side Includes: (2007). http://www.esi.org/
[20] Eiriniaki, M., Vazirgiannis, M.: Web mining for Web personalization. ACM Transactions on Internet Technology **3**(1) (2003)
[21] Flesca, S., Greco, S., Tagarelli, A., Zumpano, E.: Mining user preferences, page content and usage to personalize Website navigation. World Wide Web **8**(3), 317–345 (2005)
[22] Gao, L., Dahlin, M., Nayate, A., Zheng, J., Iyengar, A.: Improving availability and performance with application-specific data replication. IEEE Transactions on Knowledge and Data Engineering **6**(1), 106–120 (2005)
[23] Gray, J., Helland, P., O'Neil, P., Shasha, D.: The dangers of replication and a solution. In: Proc. of ACM SIGMOD Int'l Conf. on Management of Data, pp. 173–182 (1996)
[24] Groothuyse, T., Sivasubramanian, S., Pierre, G.: GlobeTP: Template-based database replication for scalable Web applications. In: Proc. of 2007 World Wide Web Conf. (2007)
[25] Guo, H., Chen, S., Xiao, Z., Zhang, X.: DISC: Dynamic interleaved segment caching for interactive streaming. In: Proc. of the 25th International Conference on Distributed Computing

Systems (2005)
[26] Guo, L., Chen, S., Xiao, Z., Zhang, X.: Analysis of multimedia workloads with implications for Internet streaming. In: Proc. of 14th Int'l World Wide Web Conf. (2005)
[27] Ho Ha, S.: Helping online customers decide through Web personalization. IEEE Intelligent systems **17**(6) (2002)
[28] IBM WebSphere Edge Server: (2007). `http://www-3.ibm.com/software/Webservers/edgeserver/`
[29] Iyengar, A., Ramasvamy, L., Schroeder, B.: Techniques for efficiently serving and caching dynamic Web content. In: S. Chanson, X. Tang, J. Xu (eds.) Web Content Delivery. Springer (2005)
[30] Larson, P., Goldstein, J., Guo, H., Zhou, J.: MTCache: Mid-tier database caching for SQL server. IEEE Data Engineering Bulletin **27**(2), 35–40 (2004)
[31] Lin, Y., Kemme, B., Patiño-Martínez, M., Jiménez-Peris, R.: Consistent data replication: Is it feasible in WANs? In: Proc. of Europar Conf. (2005)
[32] Olston, C., Manjhi, A., Garrod, C., Ailamaki, A., Maggs, B., Mowry, T.: A scalability service for dynamic Web applications. In: Proc. of Innovative Data Systems Research, pp. 56–69. Asilomar, CA (2005)
[33] Pacifici, G., Spreitzer, M., Tantawi, A., Youssef, A.: Performance management of cluster based Web services. IEEE Journal on Selected Areas in Communications **23**, 2333–2343 (2005)
[34] Patiño-Martínez, M., Jiménez-Peris, R., Kemme, B., Alonso, G.: Consistent database replication at the middleware level. ACM Transactions on Computer Systems **23**(4), 1–49 (2005)
[35] Plattner, C., Alonso, G.: Ganymed: Scalable replication for transactional Web applications. In: Proc. of ACM/IFIP/USENIX Int'l Middleware Conf. Toronto, Canada (2004)
[36] Rabinovich, M., Spatscheck, O.: Web Caching and Replication. Addison Wesley (2002)
[37] Rabinovich, M., Xiao, Z., Aggarwal, A.: Computing on the edge: A platform for replicating Internet applications. In: Proc. of 8th Int'l Workshop on Web Content Caching and Distribution. Hawthorne, NY (2003)
[38] Ramaswamy, L., Iyengar, A., Liu, L., Douglis, F.: Automatic fragment detection in dynamic Web pages and its impact on caching. IEEE Transactions on Knowledge and Data Engineering **17**(6), 859–874 (2005)
[39] Ramaswamy, L., Liu, L., Iyengar, A.: Scalable delivery of dynamic content using a cooperative edge cache grid. IEEE Transactions on Knowledge and Data Engineering **19**(5), 614–630 (2007)
[40] Rilling, L., Sivasubramanian, S., Pierre, G.: High availability and scalability support for Web applications. In: Proc. of 2007 IEEE/JSP Int'l Symp. on Applications and the Internet. Washington, DC (2007)
[41] Shi, W., Karamcheti, V.: Conca: An architecture for consistent nomadic content access. In: Proc. of Workshop on Caching, Coherence, and Consistency. Sorrento, Italy (2001)
[42] Shi, W., Shah, K., Mao, Y., Chaudhary, V.: Tuxedo: A peer-to-peer caching system. In: Proc. of 2003 Int'l Conf. on Parallel and Distributed Processing Techniques and Applications (2003)
[43] Sivasubramanian, S., Alonso, G., Pierre, G., van Steen, M.: GlobeDB: Autonomic data replication for Web applications. In: Proc. of 14th Int'l World Wide Web Conf. Chiba, Japan (2005)
[44] Sivasubramanian, S., Pierre, G., van Steen, M., Alonso, G.: Analysis of caching and replication strategies for Web applications. IEEE Internet Computing **11**(1), 60–66 (2007)
[45] Sivasubramanian, S., Szymaniak, M., Pierre, G., van Steen, M.: Replication for Web hosting systems. ACM Computing Surveys **36**(3) (2004)
[46] Szymaniak, M., Pierre, G., van Steen, M.: Latency-driven replica placement. IPSJ **47**(8) (2006)
[47] The Apache Cocoon project (2007). `http://cocoon.apache.org/`
[48] Tolia, N., Satyanarayanan, M.: Consistency-preserving caching of dynamic database content. In: Proc. of 16th Int'l World Wide Web Conf., pp. 311–320 (2007)
[49] Vakali, A., Pallis, G.: Content delivery networks: Status and trends. IEEE Internet Computing **7**(6) (2003)
[50] Williams, A., Arlitt, M., Williamson, C., Barker, K.: Web workload characterization: Ten years

later. In: S. Chanson, X. Tang, J. Xu (eds.) Web Content Delivery. Springer (2005)
[51] Yuan, C., Chen, Y., Zhang, Z.: Evaluation of edge caching/offoading for dynamic content delivery. IEEE Transactions on Knowledge and Data Engineering **16**(11) (2004)
[52] Zhao, W., Schulzrinne, H.: DotSlash: Handling Web hotspots at dynamic content Web sites. In: Proc. of IEEE Global Internet Symposium. Miami, FL (2005)
[53] Zhao, W., Schulzrinne, H.: Enabling on-demand query result caching in DotSlash for handling Web hotspots effectively. In: Proc. of Int'l Workshop on Hot Topics in Web Systems and Technologies. Boston, MA (2006)

第 5 章　CDN 模拟框架的缓存技术

Konstantinos Stamos，George Pallis 和 Athena Vakali

5.1　引言

很明显，在新的 Web 时代，人们最关注的不再是仅由文本和图片构成的静态页面，而是内容的数量和服务的可用性。就内容分发而言，大多数服务提供商更关注服务的质量。在这种情况下，代理服务器和内容分发网作为两种不同的技术解决这个问题，其共同目标是使内容贴近用户及减少了响应时间。

上述两种技术具有不同的优缺点。CDN 的主要特点是能够鲁棒地对大量的请求和内容提供服务。两者的主要缺点是，由于复制和分发的成本，内容副本的放置在大多数情况下都采用静态的方式，代理缓存服务器（surrogate server）不得不保存一些冗余的、可能过时的或用户不需要的内容，这就造成存储资源的利用无法实现最优化。另外，代理服务器（proxy server）根据不同的用户访问模式，使用缓存置换算法调整对内容的缓存。但是，代理服务器不能很好地应对大量的数据和庞大的用户群。为了结合两者的优点，一些近期的研究工作[2, 20, 29, 30]对缓存方法进行了探索。随着新的缓存方法的出现，CDN 明显需要一个用于性能评估和压力测试的平台。这个平台应该提供一个联网环境，其中包含 CDN 组件、客户端、流量以及对部署缓存策略的足够支持。

虽然理想情况下应该是在实际的网络和 CDN 中验证提出的缓存策略，但这并不总是可行和合适的。从零开始构建一个真实的 CDN 环境是不现实的，因为它需要高昂的建设成本。不仅如此，其配置工作也是极其烦琐的，涉及流量模式、链路的速度、网络拓扑和协议等很多参数。增加一个新的缓存策略需要对其运行环境中的各种网络资源进行大规模的修改。另外，商业 CDN 的私有性通常也很难用于研究目的。最后，在一个真实世界的框架中进行实验也不那么简单，因为其中存在着很多不可控的因素（如随机噪声和外部网络流量），这都会使实验结果不可再现。

为了克服现实世界模型所带来的困难，可以建立仿真模型。仿真模型在这里是指支持 Web 缓存的 CDN，模型本身必须实现下面两者的平衡，即精确地表达现实世界的模型和合理的资源管理（执行时间和内存占用）。另外，该模型应该提供一个基本的平台，可以方便地加入 CDN 部件、客户端、流量、服务、内容的类型，特别是缓存策略。缓存策略在配置和多样性上存在着各种可能的情况，大量不同的实现方案都呈现为一个树状结构。因此，最好的选择是采用一个开放的架构，对被模拟的各个实体进行合理的抽象。

目前，CDN 的模拟环境不多，相关研究人员也没有一个可供参考的标准路线图

（roadmap）去设计和实施这样一个复杂的环境。这些困难表明了开发一个在大多数情况下可用的和开放的 CDN 模拟环境的必要性，本章的写作也源于此。具体而言，本章的主要贡献是：

1）为 CDN 环境下的 Web 缓存问题提供充分的背景。

2）对于一个支持 Web 缓存的 CDN，确定其模拟需求。

3）对在一个实际的 CDN 模拟环境中验证各种缓存策略的仿真过程进行分析和建模。

4）为那些希望了解这个模拟框架的相关性能问题的研究人员提供一个路线图。

总之，本章的主要目的是提供一套可靠的设计方法，以及分享具体实践过程中的经验，同时涵盖一个 CDN 模拟框架中与 Web 缓存有关的大部分问题。

本章的其余内容安排如下：首先提出在 Web 中通过 CDN 和代理服务器进行内容分发时的各种相关问题；其次，对集成 CDN 与代理服务器的缓存特性的可能性进行验证；随后，提供动态内容和若干技术的一个分类；接下来，提供对缓存一致性问题的解决方案；最后，对如何在一个模拟环境中对提出的缓存策略进行建模和实现进行深入的分析。

5.2 Web 内容分发

以合理的效率和成本向互联网用户分发信息是一个具有挑战性的问题。Web 数据缓存和复制技术已成为解决这一问题的关键，因为它们能够提供扩展性更高的解决方案[25]。Web 缓存主要由代理服务器实现，其中内容的复制是其主要目标。大致地说，Web 缓存和内容复制的目的是将工作负载从处于过载状态的内容提供商那里转移出去，由中间环节（CDN 服务器或代理服务器）对用户的请求进行服务。互联网服务提供商（ISP）使用代理来存储最常使用或最近使用的内容。同时，Web 内容提供商可以与一个 CDN 提供商（如 Akamai）签订协议，由 CDN 服务器提供其站点的内容。在下面各节中，将对 Web 中上述两个媒介网络的主要特性进行介绍。

5.2.1 代理服务器

代理服务器由 ISP 部署在互联网中，用来应对不断增长的 Web 流量以及优化 Web 上的内容分发[33]。特别地，代理服务器充当了用户和内容提供商之间的中介，通过本地存储的内容对用户的请求进行服务。用户与代理服务器建立连接，当收到每个请求时，代理服务器首先确定被请求的对象在其本地是否存在一个有效的副本。如果该副本存在且内容没有过期，则视为一次缓存命中（cache hit），否则代理服务器必须替用户将该请求转发出去。如果用户请求的对象在本地没有满足要求的副本，那么在接收到这个新对象后，代理服务器将该对象发送至终端用户，并在其本地建立一份副本。

因此，对象在中间节点上的缓存降低了带宽消耗、网络拥堵和网络流量。同时，

由于被缓存的对象由代理服务器分发，因而减少了外部延时（将对象从源服务器传输到代理服务器的时间）。最后，代理服务器缓存提高了容错能力，因为即使在远程服务器不可用或无法访问时用户也可以获得被缓存内容的一个副本。

另外，使用共享的代理缓存有三个明显的缺陷：首先，如果缓存中的副本没有正确更新，用户可能会收到过期的数据，而且随着用户数的增加，内容提供商一般会成为瓶颈；其次，缓存在瞬时拥塞时不能提高可用性；第三，缓存服务器的系统资源（内存空间、磁盘容量、I/O 带宽、处理能力和联网资源）是有限的。

产生上述问题的原因在于，代理服务器一直以来是被设计用于局部性的工作。当代理服务器不能满足用户请求时，应该从 Web 内容提供商处获取内容。然而，这可能会导致拒绝服务（DoS），因为 Web 内容提供商不能应对庞大数量的请求（每个 Web 内容提供商提供的 HTTP 连接数是有限的）。此外，Web 内容提供商和代理服务器之间的通信可能会导致延时的增加。例如，假设一位来自澳大利亚的用户请求一个网页，其 Web 内容提供商位于美国，在这种情况下会建立大量的 TCP 连接，以实现内容提供商和代理服务器之间的通信。

5.2.2 内容分发网

图 5.1 显示的是如何在 Web 上通过代理服务器和 CDN 进行内容的分发。当缓存没有命中时，代理服务器与 CDN 通信以得到被请求的内容。具体地说，CDN 提供持有 Web 服务器副本的多个接入点（称为代理缓存服务器），其中存储着相同内容的副

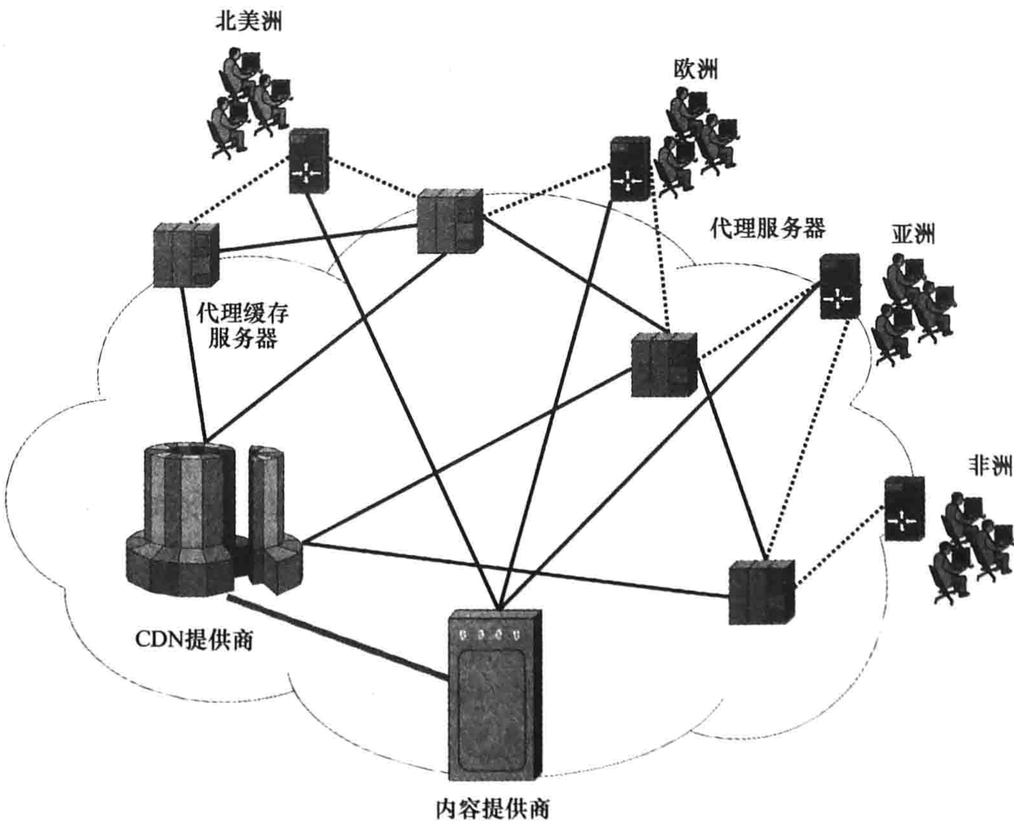


图 5.1 Web 上的内容分发

本。CDN 根据用户和被请求内容的相关信息，将用户的请求“路由”到最合适的站点。CDN 的用户是那些希望将其内容分发给在地理上广泛分布的大量终端用户的组织。CDN 在全球分布的基础上通过多个网络提供商，将其代理缓存服务器并行部署在若干战略数据中心内。表 5.1 总结了代理服务器和 CDN 之间的主要区别。本书第 2 章有一个 CDN 的全面分类，涉及组织结构、内容分发机制、请求重定向技术和性能测量方法。

表 5.1　代理服务器与 CDN 的比较

特性	代理服务器	CDN
关键技术	Web 缓存	内容复制
被缓存的内容	动态变化；由 ISP 的用户请求	由受到 CDN 支持的内容提供商预定义
可扩展性	低	高
性能	难以应对瞬时拥塞	稳定；适于资源饥渴的应用（如流媒体）

5.3　CDN 中新兴的 Web 数据缓存技术

CDN 管理着分布式全局信息资源，这些资源涉及一系列的应用。用户与公司、组织、政府机构、教育或协作环境之间（或在其内部）进行交互。CDN 之所以受到青睐，是由于它能够在全球范围内有效地分发动态、分布式、异构和非结构化数据。因此，为了改善信息在互联网上的分发，必须为 CDN 开发各种 Web 数据缓存技术和机制。

5.3.1　CDN 缓存

正如前面提到的，Web 缓存和内容复制已发展为两大类方法，以满足用户不断增加的需求。

1. Web 缓存方法

代理服务器将 Web 对象存储到自己的缓存。不过，缓存哪些对象由一个缓存置换策略确定。缓存置换策略决定哪些对象会从缓存中移出，以给新的对象提供存储位置。在这种策略下，每个对象由一个“值”来表征，即所谓的缓存效用值（Cache Utility Value，CUV）。CUV 值最小的对象在需要时将被首先清除出缓存。Podlipnig 等人在参考文献 [23] 中对现有的缓存置换策略进行了深入的总结。

2. 内容复制方法

代理缓存服务器为内容提供商保存 Web 对象的副本。与代理服务器相反，CDN 中复制的内容一直保持静态不变。

然而，CDN 的内容复制方法具有固有的局限性，主要是 CDN 设施不能对复制的内容提供有效的管理。另外，副本置换在大多数时间内是静态的。内容的静态性导致

了存储空间使用的低效，因为经过一段时间后代理缓存服务器的缓存中存放的很可能是没用的对象。因此，如果用户访问模式发生改变，那么代理缓存服务器中存放的副本将不能满足用户的要求。

为了解决上述问题，一个办法是将缓存和复制这两种策略整合到代理缓存服务器的存储空间上。Stamos 等人[30]的实验结果表明，一个综合的策略优于单独使用缓存或静态复制策略。

下面给出综合策略的规范定义。假设一个网站 W 与 CDN 提供商签订协议。网站有 N 个对象，起初只位于内容提供商处（CDN 外面），则 W 的总大小 W^S 由式（5.1）给出。

$$W^S = \sum_{k=1}^{N} U_k^S \tag{5.1}$$

式中，U_k^S 是第 k 个对象的大小（$1 \leqslant k \leqslant N$）。

设 CDN 的代理缓存服务器有 M 个。每个代理缓存服务器 M_i 的缓存总容量为 $M_i^S(1 \leqslant i \leqslant M)$，用于复制 W 的内容，而内容原件位于内容提供商处。为简单起见，假设每个代理缓存服务器的缓存容量相同，均为 M^S 且不包含其他 Web 站点的内容。

如图 5.2 所示，代理缓存服务器可以划分为两个分区。

1. 静态缓存分区

该缓存分区用于静态内容的复制，其大小定义为 M^S 的一个百分比 $r(r \in [0, 1])$。因此，位于代理缓存服务器静态缓存中的被复制对象服从约束条件（5.2）。

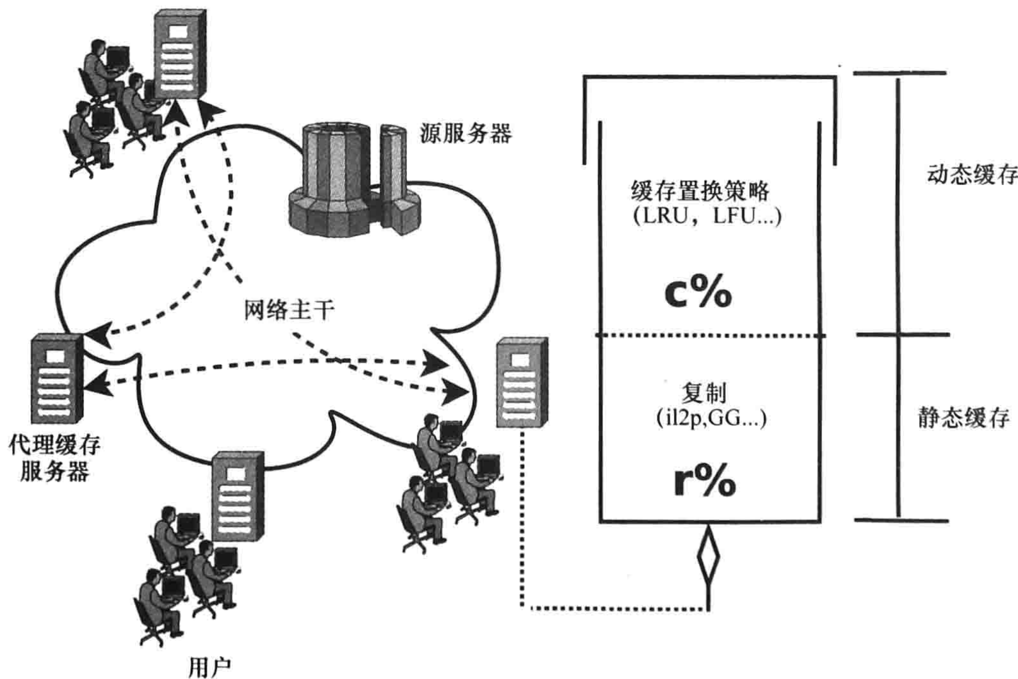


图 5.2 CDN 的集成缓存

$$\sum_{k=1}^{N} (f_{ik} U_k^S) \leqslant rM^S \tag{5.2}$$

式中，f_{ik}表示对象 k 是否存在于代理缓存服务器 i 的缓存中，即$f_{ik}=1$ 表示第 k 个对象在第 i 个代理服务器，否则$f_{ik}=0$。静态缓存中的内容取决于所采用的内容复制算法。参考文献［12，19，21，32，37］中给出了各种内容复制算法。Kangasharju 等人[12]使用四种启发式方法：随机、受欢迎度、贪心、全局贪心。实验结果表明，全局贪心算法胜过所有其他办法。然而，贪心算法的高复杂性使其在实际应用中不具备可行性。Tse[32]从另一个角度研究了内容置换的问题，提出了一系列的贪心算法，在对代理缓存服务器的负载和大小进行均衡的时候执行置换操作。Zhuo 等人也提出了一个非常类似的方法[37]。Pallis 等人[21]提出了一种自调节、与参数无关的算法（称为 Lat-cdn），该算法基于网络延时来置换 CDN 代理缓存服务器中的对象。在参考文献［19］中，Pallis 等人将内容置换问题划分为两个子问题：一是定义延时最小的“对象—代理缓存服务器”对，二是根据用户的工作负载决定复制哪些对象，这种方法被称为 il2p。

2. 动态缓存分区

该分区由使用缓存置换策略的 Web 缓存支配。为了规范地定义动态缓存分区，动态缓存的大小定义为 M^S 的一个百分比 $c(c\in[0,1])$。具体地说，被存储对象的存储容量服从约束条件（5.3）。

$$\sum_{k=1}^{N}(f_{ik}U_k^S)\leqslant cM^S \tag{5.3}$$

初始情况下动态缓存为空，这是因为其存储的内容是在运行期间根据所选的缓存置换策略填入的。代理缓存服务器的动态缓存分区内存储的是 CUV 值最高的那些副本。除了传统的置换策略（如 LRU、LFU），Aioffi 等人[1]使用在线启发式算法决定是否加入新的内容副本或删除现有的副本。该算法（称为在线 MDCDN）是一种基于统计预测的方法，称为双指数平滑（DES）。考虑到用户需求的变化，MDCDN 预测未来对每个代理缓存服务器的需求，以此确定被缓存对象的 CUV 值。在 Chen 等人[6]提出的方法中，使用一个应用级组播树作为每个代理缓存服务的缓存置换策略。Presti 等人[24]使用一个非线性整数规划方法确定副本的 CUV 值。Bartolini 等人[3]采用一个半马尔科夫决策过程决定增加一个新的副本还是删除一个现有的副本。

根据上述的缓存分割策略，百分比（r，c）需服从式（5.4）。

$$r+c=1 \tag{5.4}$$

这种方法面临的挑战，是如何确定代理缓存服务器用于缓存和复制的空间大小，即确定百分比（r，c）。考虑到这是一个 NP 完全问题，一些启发式方法可以有效地整合代理缓存服务器的静态和动态缓存。Bakiras 等人[2]提出的一个贪心混合算法将一个 LRU 缓存置换策略与静态内容复制结合起来。具体地说，最初每个代理缓存服务器的存储容量保留给 Web 缓存使用，在每一步迭代中对象根据收益值最大化的原则被放置到一个代理缓存服务器。在决定是否将一个静态内容放置在代理缓存服务器时，需要判断这样做是否可以使响应时间最优。根据这个判断条件，混合算法逐步将静态内容填入代理缓存服务器的缓存。Stamos 等人[29]提出了一个置换相似性方法（SRC），用来评价 Web 缓存与内容复制的综合水平。该办法使用一个相似性测度来

确定代理缓存服务器实现缓存和复制所需的容量大小。Pallis 等人[20]使用 logistic sigmoid 函数将代理服务器的缓存分为两部分，在他们的方法（称 R－P）中根据质量值对副本进行分类，该指标值表示用户对这个副本的兴趣，如果有兴趣则该值增大，反之减少。

5.3.2 动态内容的缓存

根据 Web 对象发生变化的频度及可预测性，动态内容可分为三类，如图 5.3 所示。内容提供商每隔特定时间进行一次更新的对象属于“定期更新”类，如一个每 5min 更新一次的新闻网页。第二种是“按需更新”类，该类中的对象在需要的时候创建，且随用户的不同可能会有不同的属性（如查询表）。最后一种是“更新不可预测”类，指那些无法预测其变化的对象。定期更新和更新不可预测这两类对象可以被缓存，而按需更新的对象是不可缓存的。

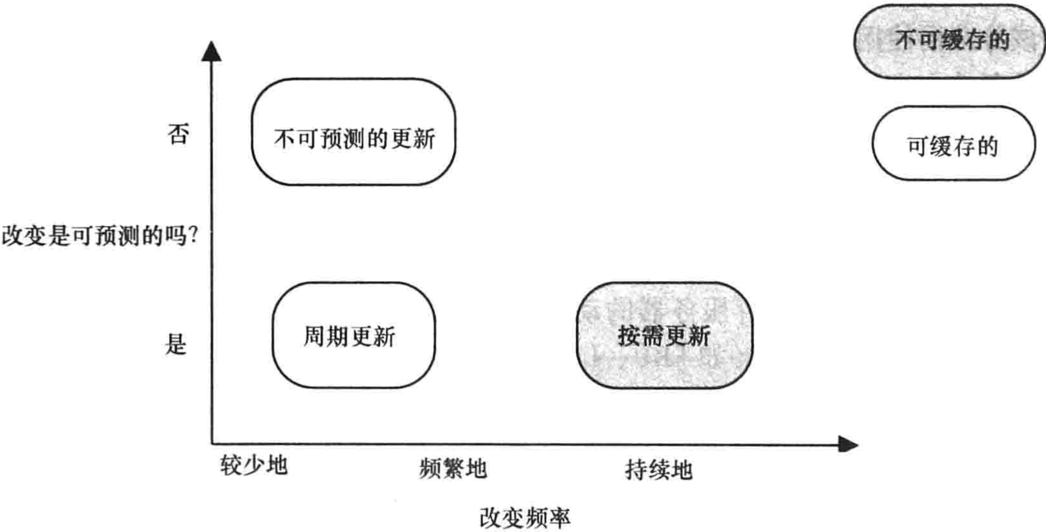


图 5.3 动态内容的分类

由于 Web 中的动态数据越来越多，如何高效地将动态内容分发到终端用户是一个重要的问题。研究人员提出了各种缓存技术以加速动态内容的分发[5,27]。当前 Web 应用通常会产生具有复杂布局的异构内容，所以基于片段的缓存是一种加速 Web 应用的有效技术。

一个片段可以被定义为网页的一部分，具有区别于页面其他部分的特定主题或功能，被独立地存储在内容提供商或代理缓存服务器上。Challenger 等人[5]使用对象依赖图表示页面与片段之间的关系。

基于片段的方法已被一些商业 CDN 提供商采用。例如，Akamai 的 EdgeSuite 网络采用了一个基于片段的策略，其中使用了被 WWW 业界认可的 ESI 规范。具体地说，ESI 规范定义了一种基于 XML 的标记语言，用于定义模板和识别页面中的片段。IBM 的 Websphere[5]也使用了一个基于片段的策略，其中页面被分解为一个由若干复杂、原子级片段构成的金字塔结构。

基于片段的方法不能有效地应用于“按需更新”类对象，因为这些对象不能被缓存。基于片段的方法适用于请求的时间局部性较高且数据库更新很少的情况，如果没有这种行为特性，那么就需要采用更复杂的技术[28]。因此，一种方法是在代理缓存服务器上复制整个应用代码的副本[26]，每个代理缓存服务器可能连接到一个中央数据库，将所有对数据库的操作请求转发给内容提供商。虽然这种技术允许通过分布式计算的方式生成网页，但会受到每个查询的延时及源数据库吞吐量瓶颈的限制[28]。为了解决这个问题，另一种方法是保存数据库的部分副本（称为内容已知缓存方法），编程人员在开发一个应用时可以复制最适于这个应用的数据。只要选定的策略与该应用相适应，该方法就可以显著地提高性能和可用性[28]。然而，这在具体实现时是相当困难的，因为它要求应用的开发人员在容错性和缓存弱一致性等方面具有很深入的理解。在这种情况下，可以使用另一种技术（称为内容未知缓存），用来在代理缓存服务器中对数据库操作的结果进行缓存。当数据库发生更新时，必须维持被缓存结果的一致性。因为许多对数据库的请求可以在本地做出响应，所以这种技术可以减少数据库操作结果的传输延时；而由于转发给内容提供商的请求变得更少了，因此整个系统的总吞吐量也有所增大[28]。

5.3.3　缓存的一致性机制

考虑到 Web 内容的动态性，维护内容的一致性成为必须解决的一个重要问题[36]。为了防止失效的内容传输给最终用户，代理缓存服务器必须确保本地的缓存数据与存储在服务器上的数据是一致的。CDN 使用哪种一致性机制以及实现到什么程度，取决于被缓存数据的特性。因此，CDN 应该采用合适的机制确保其副本与内容提供商一致。

代理服务器的一致性维护问题已得到了大量的研究。例如，代理服务器中广泛使用的 TTL 技术中[26]，在向代理服务器提供一个可缓存对象时，内容提供商提供一个 TTL 值，由代理服务器判断该对象在其生存期内是否有效。在 CDN 中，TTL 的概念可以用于每个代理缓存服务器，每个服务器负责维持其缓存数据的一致性。因此，每一个代理缓存服务器独立地与内容提供商进行交互，而与其他的服务器无关。然而，从内容提供商的角度来看，这种做法在一个大型网络中可能不具有可操作性。例如，一个典型的 CDN、通常由大量代理缓存服务器组成（如 Akamai 公司，这是一个处于主导地位的 CDN 提供商，在全世界拥有超过 25 000 台代理缓存服务器），内容提供商需要单独与大量的代理缓存服务器进行交互。

为了规范地定义一致性的实现程度，令 CP_k^t 和 S_k^t 分别表示 t 时刻对象 k 在内容提供商和代理缓存服务器处的版本，这时一个对象 k 的一致性可以有以下几种实现级别：

1）与内容提供商处的版本保持强一致（strongly consistent）。对象 k 在代理服务器上的版本始终与内容提供商的最新版本一致，即对于任意的 t，$CP_k^t = S_k^t$。强一致性忽略了数据更新的信息传到代理缓存服务器过程中引入的网络延时。

2）与内容提供商处的版本保持 Δ 一致（delta consistent）。当内容提供商修改对象内容后，需在 Δ 个时间单位内对代理缓存服务器上的版本进行更新，其中 Δ 是一

个可配置的参数，即对于任意 t，存在 τ（$0 \leq \tau \leq \Delta$），有 $CP_k^{t-\tau} = S_k^t$。

3）与内容提供商处的版本保持弱一致（weak consistent）。代理缓存服务器上的版本并不总是与内容提供商保持一致。

一致性程度也可以对多对象进行定义，称为相互一致性。为了规范地定义这个一致性程度，考虑两个相互关联的对象 a 和 b。对于对象 a 和 b 在时间 t 的缓存版本（分别为 S_a^t 和 S_b^t），定义二者在时域上相互一致（M_t 一致）的条件为：若 $CP_a^t = S_a^{t_1}$ 和 $CP_b^t = S_b^{t_2}$ 则 $|t_1 - t_2| \leq \delta$，其中 δ 是对一致性偏差的容忍度，$\delta = 0$ 表示在过去的某些时刻两个对象同时存在于内容提供商。

目前，存在着各种机制可以提供高效的缓存一致性[16, 17, 31]。这些机制可以大致分为以下几类：

1）服务器驱动的一致性（也称为基于服务器的失效性）。一旦内容发生变化，内容提供商就通知代理缓存服务器。因为只有当对象被修改时信息才发送，所以这种方法大大降低了内容提供商和代理缓存服务器之间控制信息的数量。然而，这会降低内容分发时分布式网络的使用效率，也降低了代理缓存服务器一致性管理的效率，因为这时内容提供商就需要为所有持有对象的代理缓存服务器维护一个表。最近出现的几个新协议使用基于服务器的失效性来提供一致性机制。Web 缓存失效协议（WCIP）[14] 使用应用级组播来发送服务器的失效性信息。Web 内容分发协议（WCDP）[31] 是另一种支持一致性的方法，内容提供商使用该方法可以动态控制对象更新信息的传播和可视性，WCDP 能够支持不同级别的一致性（即强一致性、Δ 一致性、弱一致性和相互一致性）。

2）客户端驱动的一致性（也称为客户端轮询）。内容提供商的对象无论什么时候发生变化，其最新版本都会随时传递给所有的代理缓存服务器，其优势是内容提供商不需要维护任何列表。然而，如果对象被更新的频率大于被访问的频率，那么这种做法可能会产生大量不必要的流量。Mikhailov 和 Wills[16] 提出了一个称为 MONARCH（用装配、关联和变化特性来管理网络对象）的方法，可以通过从感兴趣的网站上收集内容的快照来保证缓存的一致性。这些内容被输入到一个仿真器中，对一系列访问模式下的缓存一致性进行评估。

3）租期方法。该方法给每个对象赋予一个租期。租期是一个时间段，其长度表示当对象发生修改时内容提供商在该时间段内同意通知代理缓存服务器。租期期满时，代理缓存服务器必须发出一个信息，要求续订租期。这种方法是服务器驱动和客户端驱动两种一致性方法的结合。如果租期是 0，则退化为纯客户端驱动的方式；而如果租期为无穷大，则退化为纯服务器驱动的方式。租期的概念首次用于解决缓存一致性问题是在分布式文件系统中[10]，而首次在 Web 代理缓存上的应用则出现在参考文献［15］中。Duvvuri 等人[8] 对 HTTP 进行了扩展，使之包含租期的概念。Ninan 等人[17] 针对 CDN 提出了租期方法的一个变种，称为协作租期，其中使用了 Δ 一致性的思想，也就是说，要求一个对象的缓存版本相对于它的服务器版本在 Δ 的时间单位内是有效的，Δ 值决定了这个担保的强度，越大则一致性就越弱。

5.4　CDNsim 的缓存技术

本节使用一个实际的 CDN 模拟器来介绍缓存置换策略，其中将 CDNsim㊀作为主要模型。首先，介绍这种模拟环境的必要性以及其他一些模拟方法；其次，从科学问题和资源需求的角度定义一个被模拟缓存的需求；最后，就这种缓存框架在实际开发时的一些相关问题进行讨论。

5.4.1　CDN 模拟环境的需要

目前，存在一些私有和学术性质的 CDN 用于研究和日常应用，如 CoDeeN[18]、Coral[9] 和 Globule[22]。但是，由于现在面临的是实际网络，所以实验不可能重现。而且，新策略的执行和评估也很难，因为这需要对整个系统进行大规模的改造。因此，为了进行实验有必要进行模拟环境的搭建。在这个方向上，已经有了一些模拟 CDN 的具体实现[4,7,12,34]，可以满足单个研究工作的需求。但是，这些模拟 CDN 大多没有考虑到几个关键因素，如网络中可能出现的瓶颈以及每个网络资源（如路由器、代理服务器）提供的会话数量，而且它们也未考虑到 TCP/IP。除此之外，这些模拟环境最大的缺点是没有用于验证 CDN 缓存技术的平台。

CDNsim 填补了这一空白，它被开发为一个通用的 CDN 模拟器。CDNsim 用 C + + 编写，可扩展且源码开放，使用 OMNET + + 和 INET 库㊁。它是一个并行离散事件跟踪驱动网络模拟包，提供库、例程及网页内容分发的接口。CDNsim 对一个 CDN 进行了建模，该 CDN 包括了基本的网络要素，如用户、代理缓存服务器、内容提供商和路由器。通过对 TCP/IP 的模拟，该模型考虑到了互联网设施的特性。CDNsim 的设计支持对大范围内 CDN 服务的研究，它还能够模拟 P2P 网络服务以及各种网际互连的配置。下面将介绍这个模拟器的特点。

5.4.2　CDNsim 的缓存框架的要求

这里介绍 CDNsim 中代理缓存服务器的缓存规范（设计时考虑）以及解决研究和性能等方面的相关问题。对缓存设计的一个要求是要支持参考文献［2，20，29，30］中的综合缓存方法，同时也必须能够支持缓存一致性机制。此外，它需要支持视频等复杂的内容类型以及基于片段的动态内容的处理。最后，它还应该支持不可缓存的内容。

缓存置换算法的多样性导致了实现时大量的执行分支，例如 LFU 和 SIZE 利用对象的不同属性进行对象的置换。因此，有必要从各种情况中提取出公共的交集。更具体地说，给定一组基本的通用操作和内容类型，就应该能够实现上述任何一种方法。所以，必须定义一个严格的需求分析，作为编写、实现各种缓存方法的一组接口。另外，需要考虑缓存内容如何在网络上适当地发布，以允许用户和 CDN 分别实现内容的下载和管理。

㊀ http：//oswinds. csd. auth. gr/ ~ cdnsim。——原书注

㊁ http：//www. omnetpp. org/。——原书注

一个 CDN 模拟器的执行需要代理服务器的缓存具有高活动性。一个典型的模拟场景中存在若干用户，他们向 CDN 发送申请对象（网页、多媒体内容等）的请求。代理缓存服务器管理其缓存的内容，并试图对这些请求进行服务。通过增加请求的数量，所需的（主机运行模拟操作的）CPU 时间也随之增加。此外，缓存容量的增加会导致更多的对象被存储，并对内存提出更高的要求。显然，各种模拟场景和配置的不同，使缓存有可能成为性能的瓶颈。

对性能来说，我们关注的主要是在 CPU 需求的角度上优化缓存的功能。为了确定一个合适的性能阈值，缓存的操作应包括：

1）搜索（Search）。该操作在被缓存内容中寻找指定的对象。例如，一个用户请求一个网页，代理缓存服务器检查缓存中是否存储了该内容。这种操作的性能好坏取决于对内容的组织。在一般情况下搜索的复杂性应该总是优于 $O(n)$，其中 n 是指存储在缓存中的对象数。这一点至关重要，因为在模拟过程中搜索操作可能执行数百万次，而缓存中可能有数千个对象。

2）插入（Insertion）。该操作在缓存中插入一个新的对象，它负责组织好缓存以及更新其他的属性，如剩余存储空间。同样，它必须具有高于 $O(n)$ 的效率。每一个置换策略都包含插入操作。一次模拟的过程将可能执行很多次，所以这是一个至关重要的优化参数。

3）删除（Deletion）。该操作从缓存中删除一个特定的对象。每个缓存置换算法将待删除对象移除以释放存储空间，因此其性能的下限是 $O(n)$。

4）更新（Update）。更新是将对象的前一个版本删除并将更新后的版本插入。当采用缓存一致性机制时，会用到更新操作。

从文中可以看出，缓存的速度与内容在主机内存中的组织密切相关。缓存的置换算法大多包括搜索、插入、删除、更新（SIDU）等操作，因此，优化的目标就是高效的内容管理设计。

另一个被关注的次要性能是内存的优化。这一点很重要，因为模拟的执行环境需要合适的内存。因为读写速度的要求，内存中的数据绝对不能被交换到主机的硬盘上，执行性能也决不能降低。因此，需要采取保守的设计来节省内存。模拟一个包含大量代理缓存服务器、缓存和对象的大型网络，所需的内存量数以千兆字节。不过，内存优化并不是很严重的问题，因为内存容量的限制可以很容易地通过配备足够的内存来解决。

5.4.3 CDNsim 的缓存架构

为了满足上述要求，本节定义了一个架构来实现 CDNsim。同时，对 CDN 中的所有实体进行抽象，将 CDN 缓存的组织结构建模为一个三层架构，如图 5.4 所示。

第一层负责处理内容的概念问题，忽略了各种内容的特性。一般情况下，可缓存的内容被认为是原始的片段，如存储在缓存中的对象。保存对象的缓存只是一个存储的介质，保存着可用容量等最新的信息，并提供前面所述的 SIDU 接口。对象可分为两类：非易变的和易变（volatile）的对象，前者指存储在缓存中的静态对象，而后者

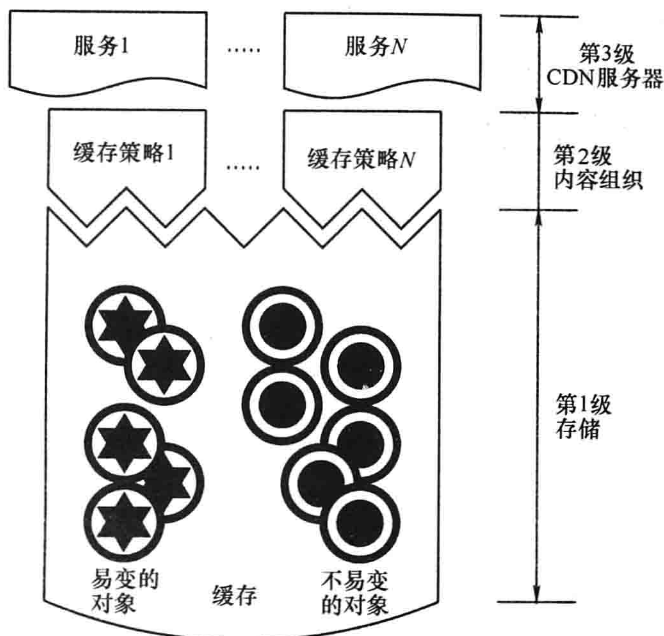


图5.4　三层缓存组织结构

是可以通过缓存置换算法移出缓存的对象。通过标记对象的方式，可以将参考文献[2，20，29，30]中提出的那些静态和动态缓存方法进行综合。

第二层负责对象的组织，并忽略每个对象的具体特性。在这种情况下，定义一组缓存置换策略来管理缓存的内容。每个策略利用一些属性来使其保持对象的有序，这些属性可以是对象的大小、TTL和最近访问时间。例如，LRU缓存策略根据对象的最近访问时间来维持一个缓冲区，以保持对象有序排列，以此作为对象置换的依据。不同于上一层使用的SIDU接口，本层引入了缓存分区的概念，即静态缓存分区和动态缓存分区。

第三层定义了一个更高级别的内容管理逻辑。相对于前两层，这一层关注的是实际的内容类型和对象的特性。每个内容类型可以表示为具有分层结构的一组对象(片段)。例如，一个动态页面代表了一组对象，表示构成页面的一组可缓存的片段。因此，在较低层次上对对象的抽象提供了解决内容多样性的统一方法。另一种高层的结构是服务，代表了在CDN层面上的任何操作。这些服务可以仅供内部使用（如代理缓存服务器之间的协作)，并只供CDN使用，否则就是公共服务，如动态网页的显示和传输。因此，可以提供适当的服务来解决缓存的一致性问题。此外，不可缓存的内容（按需更新）可以通过那些能实时创建内容的服务来处理。

表5.2总结了CDNsim中各种缓存问题和架构部件之间的对应关系。具体来说，CDNsim直接支持静态、动态和综合的缓存方案，其中每个都可以建模为缓存分区。CDNsim为任何静态副本放置算法提供了一种通用的输入，而默认情况下它支持LRU动态缓存。缓存的一致性由一组服务管理，这些服务发出各种内容更新信号并提交发生的改变。缓存一致性实现于第三层，在默认情况下CDNsim提供缓存的强一致性。复杂的内容类型如音频、视频、网页及流内容都可以通过对象的分层结构得到支持，

每个类型都可以由一组可缓存的对象代表。与一致性服务相结合，可以对动态内容的更新实现缓存，不可缓存内容可以通过第三层上的一组特定服务单独处理。

表 5.2 缓存问题与 CDNsim 架构部件之间的映射

缓存问题	架构部件	CDNsim 默认值
静态/动态缓存分区	缓存分区，易变/非易变	一般性支持/LRU
强/Δ/弱/相互一致性	服务	强
复杂内容	对象层次结构	视频、网页等
不可预测/可周期性缓存的动态内容	对象	一般性支持
按需要更新，不可缓存的动态内容	服务	不明

5.4.4 实现的问题

这里介绍 CDNsim 方案中缓存策略的具体实现，可以为开发相关软件的人员提供一些有益的参考。本节使用了一个有代表性的例子，在该例中代理缓存服务器中包含用于静态和动态内容的分区缓存[30]。缓存的静态部分使用的是一个副本置换算法，而动态部分使用 LRU。选择 LRU 是因为很多人对它很熟悉，也易于掌握。本节的目标是实现一个符合前述架构和性能需求的缓存。

1. 第一层

该层主要关心的是对类对象和缓存的实现。既然要创建一个结合静态和动态的缓存策略，那么就需要定义易变对象和非易变对象这两个类别。一个属于非易变类的对象被存储于静态缓存，而当一个对象被定义为易变对象时，它在运行时将存储于动态缓存中，并可能在将来被缓存置换策略从缓存中移除。

内存的低消耗被定义为一个次要的需求，这时需要使用一个方法来实现信息压缩和缓存置换能力的平衡。

1）完全压缩——无信息。Bloom 滤波器[13]是一个用于模拟缓存的方法。该滤波器是一个对信息打包的位数组。从缓存的角度看，上面所说的信息是指对象的标识。定义在这个位数组上的操作有读、使能和禁用，一组不同的哈希函数把每一个需要插入的标识映射到这个数组中的若干位。该方法的优点是执行速度快，因为它使用与操作，运行在本地主机的 CPU 之上，内存的需求也较小。然而，使用哈希函数会导致冲突，不同对象的标识可能对应相同的位，这会导致不准确的内容描述。而且，对象的一些相关信息无法利用，例如一些属性（如最近访问时间）不能得到存储，这就使我们不能实现 LRU 之类的缓存置换算法。

2）部分压缩——部分信息。顾名思义，这个方法仅使用部分的 Bloom 滤波器技术和对象信息[13]。在完整的对象信息表示中，每一个对象是一个实际的 C + + 对象，它存储所有必要的属性，如对象的大小和最近访问时间等。然而，Bloom 滤波器会导致信息的损失，从而不能执行 LRU。

3）无压缩——完整信息。完整的对象信息适于实现 LRU，因为对象的所有属性都是可用的。看下面这个例子：在 32 位系统中需要 16B 表示一个对象，其中标识用

4B、最近访问时间用 8B、对象大小用 4B。这样，在大小为 2GB 的内存中大致可以存储 1.3 亿个对象。因此，即使存在着其他原因的内存需求，一个标准的主机仍可以管理几百万个对象。这里，不允许使用无损的压缩方法，因为耗时的解压和再压缩会导致性能的下降。因此，建议对象与其标识之间使用一一对应的方式，且所有 SIDU 操作的算法复杂度都是 $O(1)$。

2. 第二层

该层使用缓存策略实现对内容的组织。缓存策略分为两种：静态缓存和 LRU。静态缓存的内容在默认情况下保持不变，因此被存储的对象不需要保持某种有序性。标识和对象之间使用一一对应就可以满足需要，并且 SIDU 的算法复杂度都是 $O(1)$。

而 LRU 则需要特殊处理。因为 LRU 需要将最久未被使用的对象移除，所以就要求根据最近访问时间保持对象的有序。CDNsim 使用一个缓冲区，缓冲区内的对象按最近访问时间排序，如图 5.5 所示。这个缓冲区不存储实际的对象，实际的存储由缓存负责，缓冲区只用于实现对象标识的排序。搜索操作的算法复杂度在最坏情况下为 $O(n)$，这是由于搜索对象时必须对整个缓冲区顺序扫描。插入操作的算法复杂度为 $O(1)$，因为新对象是被插入到缓冲区的开始位置（头部），如果需要的话缓冲区末端（尾部）的一些对象要被移出以释放空间。删除操作的算法复杂度为 $O(1)$，因为只需在找到被删除对象后从缓冲区中删掉它，并不需要重新排序。更新操作的复杂度也是 $O(1)$，因为一旦找到对象就可以立刻更新它，不需要重新排序。显然，搜索操作是 SIDU 操作中使用最多的，其时间复杂度在最坏情况下是 $O(n)$，可能会降低缓存的速度。然而，这完全可以忽略，原因有二：首先，借助于 CPU 内的高速缓存，性能只会受到很小的影响；其次，搜索的目标是最近使用的对象，所以只需在缓冲区的开始部分检查很少的对象。虽然搜索操作在实际中表现出令人满意的性能，但还是需要进一步改进。例如，通过维护一个额外的直接指向缓冲区内所有对象的索引，就可以获得 $O(1)$ 的算法复杂度。

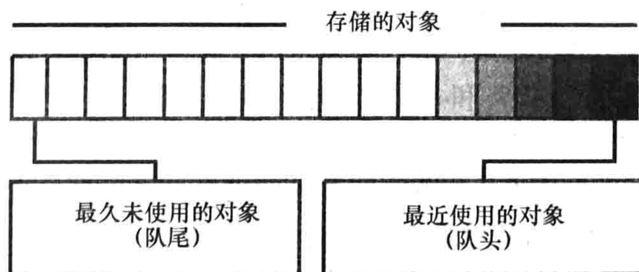


图 5.5　LRU 缓冲区

另一个问题是，当同时访问内容时可能会造成资源死锁和内容的不一致性。这个问题可以通过创建被请求内容的私有副本来解决。因此，高层的服务只处理内容的私有副本。

3. 第三层

本层中不涉及缓存问题，因为它们已经由低层解决，软件开发人员可以自由地实现那些对内容进行响应和分发的服务。

5.4.5 实验结果

参考文献［30］中的一系列的实验验证了存储器分区策略在 Web 缓存和内容复制上的效果，本节将简要地介绍该策略的一些实验结果。验证的标准包括：

1）平均响应时间。这是请求服务时间的数学期望，即所有的请求时间除以请求数，较低的值表示内容离终端用户较近。

2）响应时间 CDF。在本节的实验中，累计分布函数（CDF）表示响应时间小于或等于一个给定时间的概率。CDN 的目标是提高响应时间位于其下界附近的概率。

3）命中率。命中率定义为缓存命中数除以总的请求数。一个高的命中率代表了一个有效的缓存置换策略以及对用户的服务水平的提高和平均延时的下降。

4）字节命中率。这是以字节形式表示的命中率，定义为缓存中请求命中的字节总数除以请求的字节数。一个高的字节命中率表示网络性能的提高（即带宽的节省和低的拥塞等）。

下面对静态复制在不同综合度（r，c）时 LRU、LFU 和 SIZE 算法的性能进行了验证，其中 r 和 c 分别表示用于静态复制和 Web 缓存的存储容量的所占比例。实验中使用的静态复制算法是 il2p[19]，它使用了服务器的负载信息。具体地讲，il2p 分为两步：选择放置哪个对象以及放置到哪里。首先，根据网络延时最小的原则为每个对象选择合适的代理缓存服务器；其次，在给定的所有“对象—代理缓存服务器”备选对中，选择能产生最大效用值（取决于服务器的负载）的那一对。这个选择的实现是一个迭代的过程，直到所有的缓存填满为止。基于完整考虑，实验还包括全镜像（即整个 Web 站点被复制到缓存）和空盘（用于模拟不使用 CDN 的情况）。为了满足实验的要求，使用了 Stanford 的 Web 站点㊀。在这种情况下，利用 CDNsim 创建了一个 CDN，对数千个用户和网络资源进行模拟。

图 5.6 显示了静态复制和缓存的不同参数组合对平均请求响应时间的影响。只使用静态复制（即 $r=100\%$）时，会产生最大的平均响应时间。这个事实说明，副本放置位置固定不变时不能应对用户对内容需求的变化。从图中可以发现两个不同的性能峰值，第一个出现在 $r=80\%$ 和 $c=20\%$ 时，第二个在 $c=100\%$ 时。缓存的动态分区只增大 20%（即第一个峰值）就得到了显著的性能提高。这个现象是符合逻辑的，因为在实验中根据用户的请求为新的需要存储的内容打开一部分缓存，而同时静态副本又保持在一个 il2p 建议的有效数量。随着动态分区所占比例的增加，复制所具有的那些原本不错的性质逐渐丧失。引起这个现象的原因是，缓存置换策略可能移除了有用的内容，而根据 il2p 策略这些内容应该保持在缓存中。缓存方案的性能（第二个峰值处）表现得比综合方案（第一个峰值处）稍好一点，一个可能的原因是：将整个存储空间用于缓存置换可以使 Web 缓存方案有效地适应用户的流量模式。不仅如此，因为 Stanford 网站中的对象基本都较小，这就使性能在缓存未命中的时候只有一个小的下降。然而，缓存并不是一个完美的选择，因为这时 CDN 变成了一个代

㊀ http//www.stanford.edu/~sdkamvar/research.html。——原书注

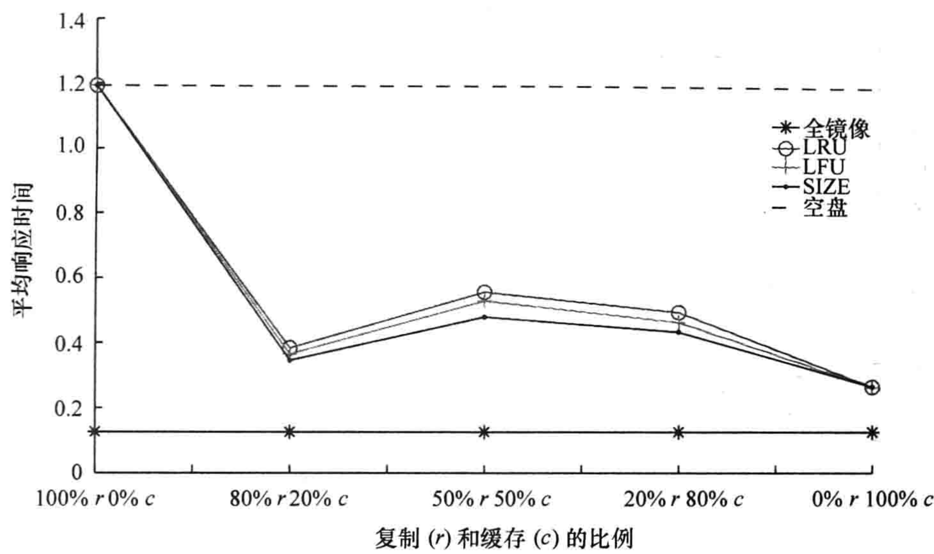


图 5.6　平均响应时间

理服务器，从而带来了一个代理服务器所具有的所有扩展性问题。另一个重要的事实是，所有的缓存置换算法都有一个相同的表现，即结合 SIZE 后性能都会变得稍好一些。SIZE 的优越之处可以被解释为，新的对象获得了更多的空间，从而提高了缓存空间的利用率。

从命中率的角度来说，在图 5.7 中可以看到与图 5.6 中相同的两个峰值。纯粹的 Web 缓存方案得出了与综合方案类似的结果，而纯静态复制则没有。在整个存储空间中固定放置副本的方案出现了低命中率的现象，这是由于冗余的内容被移除而置换却不是最优的。不能获得最优置换的原因在于：用户请求模式的改变以及它是一个 NP—完全问题。这个原因也揭示了为什么纯粹的 Web 缓存只是在性能上稍微优于综合的方案。图 5.8 显示的是字节命中率的实验，出现了与上述情况相似的结果。

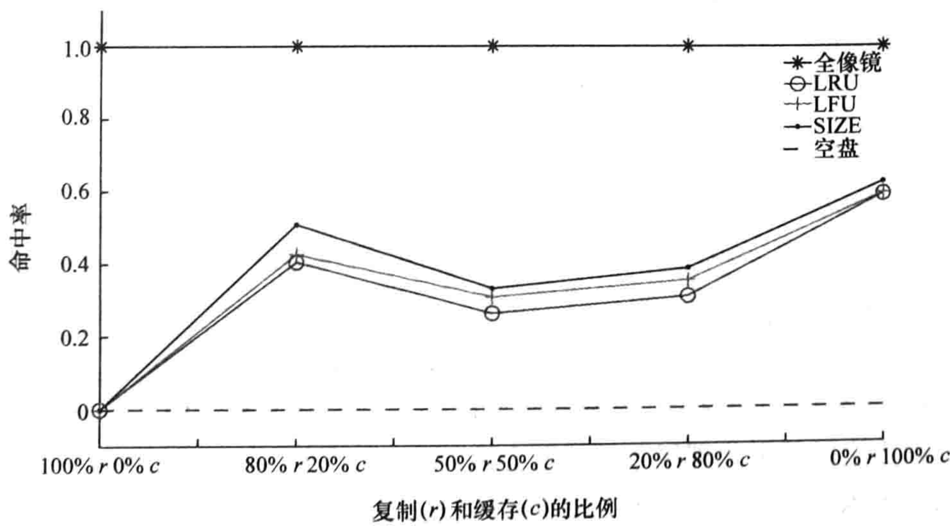


图 5.7　命中率

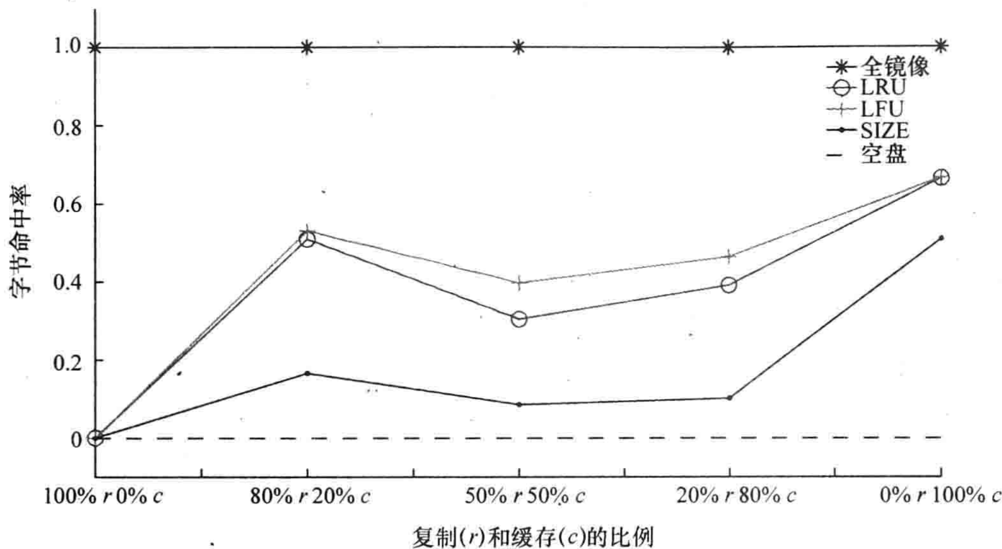


图 5.8 字节命中率

图 5.9 给出了请求服务时间分布的一个更精确的表示，读者可以注意第一个性能峰值（$r=80\%$，$c=20\%$）。忽略全镜像这种理想情况，可以看出综合方案（SIZE/i12p）具有最好的性能，取得了比其他方案更短的响应时间（其性能曲线在所有其他曲线的上面）。对实验结果的分析如下：

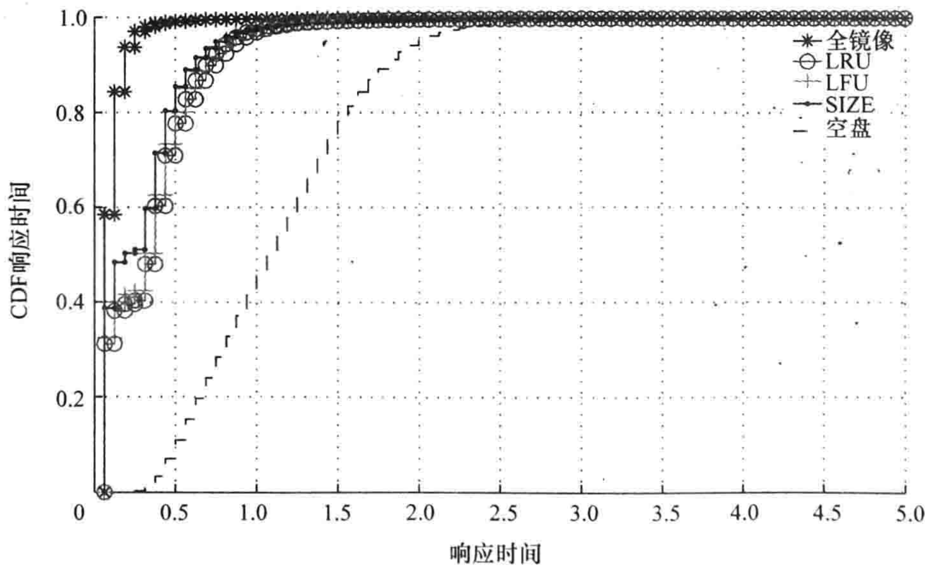


图 5.9 响应时间的累积分布函数

1）至少有一个性能上的峰值属于缓存和复制的综合方案。

2）综合方案可能由于副本放置的效率不够理想而造成性能的下降。

3）在某些特殊的缓存中，纯缓存方案的性能与综合方案一样好，但还是不建议使用它，因为它继承了代理服务器的所有缺点。

上述结果表明，通过将 Web 缓存的特性引入到 CDN 中，可以使性能具有现实的改进空间。

5.5 写给使用者

CDNsim 是一个可用于研究和商业用途的免费、开放的工具，有两个主要的用户群体可以从中受益：CDN 的开发者和对 CDN 感兴趣的软件开发人员。这节中，我们讨论怎样使用 CDNsim。

对于第一个群体来说，CDN 提供商的兴趣在于最大化其网络设施的效用。为了实现这个目标，CDN 开发者们设计了专用算法来有效地管理内容，其很自然的结果就是新产品的面世。从 CDN 的角度来看，这个新产品是一个新的内容分发服务，如视频流和大文件的分发等。虽然每个服务㊀可能在功能上互不相同，但所有服务的生命周期都存在着一系列相同的阶段。

1. 服务发布前

这个阶段涉及服务提交给用户之前的开发过程。CDNsim 可以在开发阶段使用，用于原型的设计和实现，以形成最初的产品规划。一旦完成了原型开发，可以用 CDNsim 来评估实验室条件下原型在各种网络配置和流量模式下的性能和表现。从测试和构建原型到取得一定的成熟度，CDNsim 可以显著地降低设施的投资。随后，就可以在真实的 CDN 环境中进行评估。使用 CDNsim 构建原型和测试的一个实际例子是近期由 Akamai 开发的高定义视频流㊁。

2. 服务发布后

当服务向更广泛的公众发布时，它应该已经通过了一系列的测试，而且也应该对服务的表现和性能形成一套文档化的结论。但是，由于这时产品运行在一个尚未测试过的新环境中，服务的性能表现可能会与初始的结论发生偏差。CDNsim 可以用于复现一个有问题的或意料外的环境，以帮助分析人员搞清为什么会出现那些偏差。因此，在不希望出现的局面发生之前，CDNsim 可以用做预防机制。例如，Limelight Networks 对世界杯足球赛进行一次世界范围内的播报之前，显然很有必要进行行为预测和灾难预防㊂。

3. 服务的完善期

最终，一个服务将达到某种程度的成熟性、稳定性和正确性。然而，服务的运行环境（网络配置、典型的用户群、当前的技术等）是不断变化的。有代表性的例子是互联网快速连接的增加和因可用 IP 地址不断减少导致的 IPv6 技术[11]。可以用 CDNsim 进行一个假设分析：服务是如何随着用户群体的增大而扩展的？用户连接会变得越来越快，服务和当前设施能不能跟上这一变化？CDNsim 能够通过调整相应的网络配置来验证这些问题。如果不能预测这个长期的演化趋势，就无法及时投资更新网络设施，最终导致客户的流失。

对于第二个目标群体，鼓励软件开发人员扩展现有的架构，以适应算法的最新发

㊀ 这里，“服务”这个术语一般指内容分发服务。——原书注

㊁ http：//www. akamai. com/。——原书注

㊂ http：//www. limelightnet. com/。——原书注

展趋势。估计 CDNsim 将发展为一个综合性的测试平台，其中将包含一整套用于比较和测试性能的缓存算法。不仅如此，CDNsim 可以在并行环境中运行，高性能计算的研究者可以将其作为一个在 CDN 环境中实现并行算法测试的平台。因此，有人对缓存算法的设计和实现提出了一些设想，以期利用新的多核处理器的优势和采用新的更有效的数据结构。进一步的研究方向将在下一节介绍。

5.6 未来研究方向

CDNsim 也许在内容分发领域的研究方向上提供了新的视角，下面对 CDNsim 的应用场合提出一些建议。

1. 内容分发

CDN 面临着一些问题，如：在哪儿放置代理缓存服务器？哪些内容应该转由 CDN 分发？如何实现分发的外包？对这些问题业界有着不同的看法。显然，对于 CDN 提供商来说，在这些问题上的每个决策都会产生不同的成本和局限。在 CDNsim 这个框架下，不仅能够在一个大范围内用于对各种策略进行评估，也可以用于挖掘缓存所带来的收益。

2. CDN 服务的定价

这是 CDN 提供商管理层面临的一个带有挑战性的问题。对新服务（如 EdgeSuite）的部署，总是伴随着定价和服务的问题。第 8 章探讨了一些定价的问题，并介绍了一些定价模型，使用 CDNsim 可以对它们进行验证。

3. 移动 CDN

无论在学术圈还是业界，移动无线网络的内容分发都是一个既重要又吸引人的新课题。考虑到最近在移动内容联网（如 WAP、IPv6 等）方面的进展，在无线 Web 新技术的采用方面，移动 CDN 设施可能将发挥主导作用（移动 CDN 将在第 14 章详细介绍）。CDNsim 可以用做测试平台，在移动 CDN 领域开拓新的研究思路。

4. 对等 CDN

在科研人员中，对等 CDN 的知名度正在不断上升。在对等 CDN 问题上，目前已经提出了一些方法，但有些关键问题尚待解决，包括何时使用对等及如何实现对等，第 16 章将详细讨论这些问题中的部分内容。CDNsim 可以在实际流量、工作负载和复制等条件下模拟对等 CDN 框架，同时它也可以用于对提出的对等 CDN 框架中的负载测量、请求重定向、内容复制的最佳方法和最新技术进行评估。

5. CDN 的安全性

互联网电子商务的快速发展已经引起了对 CDN 数据安全问题的很大关注[35]。在这个背景下，应该研究安全的内容分发协议以保证内容的完整性（即被未经授权的实体修改过的内容不应接受）和机密性（即分发的内容不能被未经授权的代理服务器或请求者之外的其他用户等非授权实体浏览）。CDNsim 高度的扩展性允许研究人员在其设施上完善自己的协议（如 iDeliver[35]）。

6. CDN 中的 P2P 和网格技术

既然 CDN 是复杂的大规模分布式系统，其发展可以得到 P2P 和网格等新技术的支持。在 CDN 设施上成功地开发、集成这些模型和技术将会有效地解决前述问题，同时也有助于开发更有效的 CDN。CDNsim 架构能够很容易地增强上述新技术。

5.7 结论

目前，Web 已经取得了快速的发展，从一个简单的只提供静态文本和图片的共享机制发展为可以提供多种动态和交互服务，如视频/音频会议、电子商务及远程教育等。然而，Web 的快速发展已经对建网资源和 Web 内容提供商提出了巨大的要求。从遥远的站点上获取网页时，用户常体验到很大的和无法预测的延时。CDN 设施也许能有效地解决 Web 的容量和性能问题，因此越来越多的 Web 内容提供商依靠 CDN 实现内容的分发。为了满足这些不断增长的要求，解决问题的关键在于对 CDN 内容复制的管理。具体来说，为了改善 Web 信息的分发，必须进一步研究各种 Web 数据缓存的技术和机制。

本章介绍了用于 CDN 模拟框架的各种新出现的缓存技术，研究了如何将缓存策略集成到 CDN 的设施中。另外，本章也对 CDN 的缓存一致性机制进行了全面的综述。进而，介绍了 CDN 环境中用于动态内容分发的缓存技术。最后，在 CDN 分析模拟工具 CDNsim 中对这些技术的效果进行了验证。

综上所述，CDN 的发展仍然处于一个早期阶段，其未来的发展尚无定论。目前的关键是要理解现存的与 CDN 框架有关的应用实践，以提出或预测其发展脉络。基于这个考虑，与缓存技术有关的实践似乎为 CDN 的进一步发展提供了一个有效的路线图。

参考文献

[1] Aioffi, W. M., Mateus, G. R., Almeida, J. M., Loureiro, A. A. F.: Dynamic content distribution for mobile enterprise networks. IEEE Journal on Selected Areas on Communication **23**(10) (2005)

[2] Bakiras, S., Loukopoulos, T.: Increasing the performance of cdns using replication and caching: A hybrid approach. In: IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), p. 92.2. IEEE Computer Society, Washington, DC, USA (2005)

[3] Bartolini, N., Presti, F. L., Petrioli, C.: Optimal dynamic replica placement in content delivery networks. In: 11th IEEE International Conference on Networks (ICON 2003), pp. 125–130. Sydney, Australia (2003)

[4] Bent, L., Rabinovich, M., Voelker, G. M., Xiao, Z.: Characterization of a large web site population with implications for content delivery. World Wide Web **9**(4), 505–536 (2006)

[5] Challenger, J., Dantzig, P., Iyengar, A., Witting, K.: A fragment-based approach for efficiently creating dynamic web content. ACM Transactions on Internet Technology. **5**(2), 359–389 (2005)

[6] Chen, Y., Katz, R. H., Kubiatowicz, J.: Dynamic replica placement for scalable content delivery. In: IPTPS, pp. 306–318. Cambridge, USA (2002)

[7] Chen, Y., Qiu, L., Chen, W., Nguyen, L., Katz, R. H.: Efficient and adaptive web replication using content clustering. IEEE Journal on Selected Areas in Communications **21**(6) (2003)
[8] Duvvuri, V., Shenoy, P., Tewari, R.: Adaptive leases: A strong consistency mechanism for the world wide web. IEEE Transactions on Knowledge and Data Engineering **15**(5), 1266–1276 (2003)
[9] Freedman, M. J., Freudenthal, E., Mazieres, D.: Democratizing content publication with coral. In: 1st USENIX/ACM Symposium, vol. 2004
[10] Gray, C., Cheriton, D.: Leases: an efficient fault-tolerant mechanism for distributed file cache consistency. In: SOSP '89: Proceedings of the twelfth ACM symposium on Operating systems principles, pp. 202–210. ACM, New York, NY, USA (1989). http://doi.acm.org/10.1145/74850.74870
[11] Huston, G.: Ipv4: How long do we have? The Internet Protocol Journal **6**(4) (2003)
[12] Kangasharju, J., Roberts, J. W., Ross, K. W.: Object replication strategies in content distribution networks. Computer Communications **25**(4), 376–383 (2002)
[13] Kulkarni, P., Shenoy, P. J., Gong, W.: Scalable techniques for memory-efficient cdn simulations. In: WWW, pp. 609–618 (2003)
[14] Li, D., Cao, P., Dahlin, M.: Wcip:web cache invalidation protocol. IETF Internet Draft (2000)
[15] Liu, C., Cao, P.: Maintaining strong cache consistency in the world-wide web. In: ICDCS '97: Proceedings of the 17th International Conference on Distributed Computing Systems (ICDCS '97), p. 12. IEEE Computer Society, Washington, DC, USA (1997)
[16] Mikhailov, M., Wills, C. E.: Evaluating a new approach to strong web cache consistency with snapshots of collected content. In: WWW '03: Proceedings of the 12th international conference on World Wide Web, pp. 599–608. ACM, New York, NY, USA (2003)
[17] Ninan, A., Kulkarni, P., Shenoy, P., Ramamritham, K., Tewari, R.: Cooperative leases: scalable consistency maintenance in content distribution networks. In: WWW '02: Proceedings of the 11th international conference on World Wide Web, pp. 1–12. ACM Press, New York, NY, USA (2002)
[18] Pai, V. S., Wang, L., Park, K., Pang, R., Peterson, L.: Codeen. In: Second Workshop on Hot Topics in Net-working (HotNets-II) (2003)
[19] Pallis, G., Stamos, K., Vakali, A., Katsaros, D., Sidiropoulos, A.: Replication based on objects load under a content distribution network. In: ICDEW '06: Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06). IEEE Computer Society, Atlanta, USA (2006)
[20] Pallis, G., Thomos, C., Stamos, K., Vakali, A., Andreadis, G.: Content classification for caching under cdns. In: Innovation on Information Technology. IEEE Computer Society, Dubai, United Arab Emirates (2007)
[21] Pallis, G., Vakali, A., Stamos, K., Sidiropoulos, A., Katsaros, D., Manolopoulos, Y.: A latency-based object placement approach in content distribution networks. In: Third Latin American Web Congress (LA-Web 2005), pp. 140–147. Buenos Aires, Argentina (2005)
[22] Pierre, G., van Steen, M.: Globule: a collaborative content delivery network. IEEE Communications Magazine **44**(8), 127–133 (2006)
[23] Podlipnig, S., Böszörmenyi, L.: A survey of web cache replacement strategies. ACM Computing Surveys **35**(4), 374–398 (2003)
[24] Presti, F. L., Petrioli, C., Vicari, C.: Dynamic replica placement in content delivery networks. In: 13th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2005), pp. 357–360. Atlanta, GA, USA (2005)
[25] Rabinovich, M., Spatscheck, O.: Web Caching and Replication. Addison Wesley (2002)
[26] Rabinovich, M., Xiao, Z., Douglis, F., Kalmanek, C. R.: Moving edge-side includes to the real edge – the clients. In: USENIX Symposium on Internet Technologies and Systems. Seattle, Washington, USA (2003)
[27] Ramaswamy, L., Iyengar, A., Liu, L., Douglis, F.: Automatic fragment detection in dynamic web pages and its impact on caching. IEEE Transactions on Knowledge and Data Engineering **17**(6), 859–874 (2005)
[28] Sivasubramanian, S., Pierre, G., van Steen, M., Alonso, G.: Analysis of caching and replication strategies for web applications. IEEE Internet Computing **11**(1), 60–66 (2007)
[29] Stamos, K., Pallis, G., Thomos, C., Vakali, A.: A similarity based approach for integrated web caching and content replication in cdns. In: Tenth International Database Engineering and Applications Symposium (IDEAS 2006), pp. 239–242. Delhi, India (2006)

[30] Stamos, K., Pallis, G., Vakali, A.: Integrating caching techniques on a content distribution network. In: Advances in Databases and Information Systems, 10th East European Conference, ADBIS 2006, pp. 200–215. Thessaloniki, Greece (2006)
[31] Tewari, R., Niranjan, T., Ramamurthy, S.: Wcdp: Web content distribution protocol. IETF Internet Draft (2002)
[32] Tse, S. S. H.: Approximate algorithms for document placement in distributed web servers. IEEE Transactions on Parallel and Distributed Systems **16**(6), 489–496 (2005)
[33] Vakali, A., Pallis, G.: Content delivery networks: Status and trends. IEEE Internet Computing **7**(6), 68–74 (2003)
[34] Wang, L., Pai, V. S., Peterson, L. L.: The effectiveness of request redirection on cdn robustness. In: 5th Symposium on Operating System Design and Implementation (OSDI 2002)
[35] Yao, D., Koglin, Y., Bertino, E., Tamassia, R.: Decentralized authorization and data security in web content delivery. In: SAC '07: Proceedings of the 2007 ACM symposium on Applied computing, pp. 1654–1661. ACM, New York, NY, USA (2007)
[36] Yin, J., Alvisi, L., Dahlin, M., Iyengar, A.: Engineering web cache consistency. ACM Transactions on Internet Technology **2**(3), 224–259 (2002)
[37] Zhuo, L., Wang, C. L., Lau, F. C. M.: Load balancing in distributed web server systems with partial document replication. In: 31st International Conference on Parallel Processing (ICPP), p. 305. Vancouver, Canada (2002)

第6章 动态内容的请求重定向

Supranamaya Ranjan

6.1 引言

WWW作为一种无处不在的媒介，用户对它的依赖不断增长，越是在某个Web服务不可用的时候这一点就变得越明显。而且，由于目前访问带宽比十年前快得多，用户对Web服务质量的期望值变得更高，由此也使他们越来越不能容忍网络吞吐量或响应时间的恶化。网络服务的崩溃和恶化来自两种原因的过载：一是每日时段（Time - of - Day）效应，该效应是指大多数站点的访问流量随每天的不同时段而变化，即大多数用户更多地在白天而不是夜晚访问网络。网络使用高峰时的流量有时相当于非高峰时的2～20倍[12, 13]。因此，在规划一个网站的资源时，管理员面临的难题是以满足最大使用量为设计依据还是以平均使用量为设计基础，而这两者都有其缺点。对于前者，它可以得到一个较好的响应时间，但大多数资源在非高峰期间可能一直处于使用不足的状态。而且，有时对最大流量的预测是非常困难的，这就造成分配资源的困难。对于后者，它可能在整体上具有更好的资源利用率，但在流量高峰期间Web用户可能无法获得最好的响应时间。造成崩溃的第二个原因是瞬时拥塞效应，该效应是指网站的用户数在一个未预料到的时刻或以未预料到的数量突然出现激增。一个最众所周知的例子是1999年的维多利亚秘密网络广播（Victoria Secret's Webcast），它一经推出便立即引起轰动，吸引了150万用户，远远超出了网站的吞吐能力，因此导致网站瘫痪，使得没有一个用户能够观看这场网络直播。

为了保证用户在网络过载情况下仍能得到一个满意的浏览体验，Web内容提供商越来越多地将内容放置和分发的工作转移到CDN（如Akamai[2]、Limelight Networks[3]或Mirror Image[4]等）。CDN最初的目标是降低用户访问数据时客户端与内容服务器之间的端对端（end - to - end）响应时间，CDN为此采用两种方法：一种是内容放置方法，即将内容放置在距离用户较近的地方；另一种是重定向方法，即将用户的请求转发到最合适的服务器。内容放置方法需要识别出热点（hot spot）的位置，即引发大量请求的地方[18, 32]，以便将内容复制到这些地方。重定向面临的问题则是如何选择对这些请求进行服务的最佳服务器[20, 22, 26, 28, 35, 37]。本章的目标是研究请求重定向的技术，以选择最佳的服务器。这里说是的“最佳”可以被定义为最近的服务器，或负载最小从而可用性最高的服务器，或者两者的混合。CDN采取的典型做法是将一个网站的内容复制到其镜像服务器上。而另一种方法是，大的Web内容提供商可以将它们的内容托管到全球范围内的多个服务器集群上，其中每个集群包含一组服务器，这些服务器可以就地处理一个用户的请求。在这种情况下，CDN只是提

供重定向技术，将用户请求转到最佳集群中的最佳服务器上。

一个请求应该由最佳服务器来响应，但最佳服务器的确定取决于内容的类型。一般来说，静态内容属于网络密集型，在整个时间范围内不发生变化（如图片或 pdf 文件），所以最佳服务器就应该选择离用户最近的那个。所谓最近的服务器，其定义并不统一。一种定义是距离用户的跳（hop）数最少的服务器。网络跳数可以定义为客户端与服务器之间链路上存在的路由器的个数（可以通过 traceroute 工具测量），也可以定义为客户端与服务器之间最短路由上存在的自治系统（Autonomous System，AS）的数量。最近服务器的另一种定义是在所有可达的服务器中，通向该服务器的路径中的瓶颈连接的网络带宽是最大的。最近服务器还可以根据网络延时的数学期望最小来定义，这时网络延时就是一个瓶颈连接的网络带宽和跳数的函数。

然而，随着用户浏览体验趋于个性化，内容提供商在其 Web 站点上使用的动态内容不断增多。这种动态创建的内容取决于用户的位置、以往的请求以及请求本身的特点（如最新的股票报价或拍卖竞价）。在这种情况下，一个典型的网页将包含动态和静态的内容片段。不过，动态内容对相关资源的要求与静态内容不同。由于动态内容的处理涉及创建新进程（如 CGI 脚本）或访问数据库表（如 PHP 脚本），所以相对于静态内容它更属于计算密集型。不仅如此，由于在互联网核心处的过度配置[21]，在路由器排队延时最小时，网络主干之间的延时越来越取决于光速的延时。这就意味着，为静态内容设计的服务器选择机制对动态内容来说可能不是最优的。对于静态内容来说，将请求转发到最近服务器的做法是有意义的，但对于动态内容来说最佳服务器的选择机制必须同时考虑到网络延时和服务器负载。在某些条件下将对动态内容的请求转发到最近服务器上也许是有效的，但在另外一些条件下，情况可能就不同了，因为如果能获得最小的网络总延时和最小的服务器平均处理时间，那么也许将请求转发到最远的服务器上反倒更好些。

根据一个请求在执行过程中在什么地方进行重定向的决策，请求的重定向策略可以分为客户端（client - side）重定向或服务器端（server - side）重定向两大类。在一个客户端重定向策略中，客户端被一个客户端代理服务器重定向到一个服务器。如 Akamai 的产品中，这些客户端代理服务器（也就是边缘服务器[19]）构成了一个覆盖网（overlay network），并使用 traceroute 和 ping 命令估计它们之间以及它们与内容提供商之间链路的延时。当一个客户端向域名服务器查询某个网站时，域名服务器返回给它三个通向内容托管服务器的路由：直接的、最优两跳和次优两跳。然后，客户端在这三条路由上同时开始下载，并在其中最快的那个路由上继续下载。简单地说，Akamai 首先在网络邻近度（network - proximity）的基础上选择三个服务器，随后根据这三个路由的速度差异判断出服务器的负载情况，并将其与服务器距离因素结合起来综合考虑。除了这种基于 DNS 的重定向技术，CDN 也使用 URL 的改写机制，即通过 DNS 将产生内容的站点的地址转换为可以提供内容服务的 CDN 客户端代理服务器，如 Akamai 将网址 www. foo. com/bar. gif 修改为 a1. akamai. net/www. foo. com/bar. gif。实际中，URL 的改写方式随不同 CDN 而异。

相比而言，在一个服务器端的重定向技术[34]中，使用客户端机制先将请求转发

到一个最初的服务器集群，然后集群重定向器将请求重定向到“最佳服务器”（既可以是本地的也可以是远程的集群）。本章主要讨论服务器端的动态内容重定向算法，算法可以将请求从一个过载的 Web 集群转发到其远处的一个副本。不过，读者会发现针对动态内容的大部分策略可以被扩展到其他的内容类型（如静态内容或流媒体）。因此，本章介绍了一个每请求广域重定向（per - request Wide - Area ReDirection，WARD）策略。根据被请求的服务是位于远程（通过重定向）还是本地，可以确定整个链路的延时及服务器处理的总延时是否达到了最小，而 WARD 的主要目标就是借此最小化客户端在端对端通信中的延时。特别地，客户端的请求可以先借助一个客户端重定向机制转到一个初始的 Web 集群，这个机制既可以是简单的 DNS 轮转算法⊖，也可以是复杂的服务器选择方案[37]。请求到达初始集群后，一个请求分派器使用一个基于测量的延时最小化算法来决定将请求转发到一个远程还是本地服务器。相对于其他方法（见 6.2 节），集群重定向器结合链路状况和服务器处理延时实现了总延时的最小化。

本章也提出了一个简单的解析模型，用于分析广域请求重定向方法对端对端延时的影响。这个解析模型可以对 WARD 带来的收益进行一个系统的性能评估。从这个模型可以发现，对于那些涉及动态内容的网络应用，由于对服务器负载信息误差的容忍度要远远低于对网络延时误差的容忍度，所以一个服务器选择机制必须获得精细的服务器负载信息。这说明，对于涉及动态内容的应用，一个服务器端重定向策略可以取得比客户端重定向策略更好的性能，而后者在与服务器端策略相同的粒度和相似的一般性开销时不能获得服务器的负载信息。

最后，本章介绍了一个概念验证测试平台的设计。在这个平台上，比较了 WARD 和其他的 strawman 策略，如只考虑网络延时或服务器负载的重定向方法以及使用轮转策略的重定向方法。这个测试平台模拟的 CDN 包括两个地理上很远的 Web 集群，这两个集群通过一个广域连接相连，连接的往返时间和拥塞等特性用 Nistnet[6] 进行仿真。每个 Web 集群都是多层结构，包括 Web 服务器层、请求分派层和数据库层。Web 集群上的应用是一个在线书店，使用 TPC - W 基准[9] 建模。实验结果显示，对于一个有 300 个并发用户的电子商务网站，广域重定向可以将平均响应时间从 5s 降低到 2.3s，下降了 54%。

本章余下内容安排如下：在 6.2 节给出了一个当前重定向技术的技术背景；在 6.3 节介绍了目前 Web 集群普遍使用的多层体系；6.4 节描述了在实现广域请求重定向要求下集群的系统架构；在 6.5 节，为研究该架构开发了一个排队模型；6.6 节中，对远程分派的请求比例以及不同系统模式下的平均响应时间进行了定量分析；在 6.7 节中介绍了测试平台的实现和度量；在 6.8 节给出了一些供使用者参考的设想；最后，未来的研究方向和本章的结论分别在 6.9 和 6.10 节给出。

⊖ DNS 轮转（round - robin）是一种用于实现 Web 服务器负载均衡的调度算法，在具有此功能的 DNS 服务器中，一个名字对应多个 IP 地址。当被询问到此名字时，DNS 服务器按顺序把这些 IP 地址返回给不同的用户，从而实现网络负载的均衡。——译者注

6.2 相关工作

最小化Web访问时间的方法可以分为资源管理和请求管理两类，而后者可以分为客户端重定向和服务器端重定向两类。

一个最小化Web访问时间的方法是确保集群有足够可用的资源。服务器迁移是将集群中未用或负载较轻的服务器分派给那些被用户大量访问的应用[33]。服务器迁移需要将应用的状态从一个服务器迁移到一个新的服务器，所以迁移时间以10min为数量级。因此，服务器迁移是一个在长时间（数分钟或数小时）运行中避免瓶颈的手段，可以用来应对每日时段效应等情况。重定向不仅能够解决长时间运行中的瓶颈（当然这需要付出重定向的额外开销），而且也能解决短期瓶颈（如瞬时拥塞引起的瓶颈）。服务器共享[36]与服务器迁移类似，不同之处是它只分派一部分的资源。服务器迁移和服务器共享是互不相关的请求重定向方法，建议将这两种机制融合使用。

相当一部分研究工作集中在客户端重定向机制上，如CDN请求重定向[25,37]、服务器选择技术[17,20]、缓存[27]、镜像和镜像放置[18,23]。这些技术都基于同样一个前提，即网络是最主要的瓶颈，而对动态内容的处理可以将瓶颈转移到集群中的资源上（一般是服务器的CPU）。这些方法能用于找到最佳的初始集群，WARD的由集群驱动的重定向方法在本质上结合了服务器延时和网络延时。

如果瓶颈没有找到或它随时间变化，那么一个集成了客户端和服务器端两种机制的重定位方法则是可能的和有益的。Cardellini等人[16]给出的方法就是这样的一种混合架构：其服务器端的重定向机制可以在CPU利用率超过了一个特定的阈值时使用HTTP重定向技术实现整个Web请求的重定向。他们得出的结论是，服务器端重定向应该有选择地使用。相比而言，本章将服务器端重定向视为当前和将来集群使用的一种基本的机制，本章的重定向机制不是基于阈值，而是无论CPU利用率为何值都能够得到最优的集群响应时间。不仅如此，Cardellini的设计策略孤立地看待网络邻近度和服务器负载，而本章的重定向策略将两者结合起来。最后，本章中将分派器作为集群架构中能够在任何层实现请求重定向的一个基本的构成模块，而不是等同于HTTP重定向。

参考文献［17，18，20，23，25，27，37］中的方法是为静态内容设计的，而另一些方法（如参考文献［15］及Akamai EdgeSuite[1]中使用的方法）是通过动态片段缓存解决动态内容的服务器选择问题。缓存既可以在客户端使用，也可以在服务器端使用。在客户端内容的过期时间是利用cookie来设置，而在服务器端（反向代理服务器），被缓存网页的过期与否由收到的数据库更新操作来决定。不过，这类方法有时会导致将失效的数据提交给客户端，或者增大网站开发和管理的复杂性。缓存可以对那些使用WARD的方法提供补充：如果被请求的内容不在缓存中，那么该请求可以被转发到一个能最快处理它的本地或远程的服务器。

目前CDN提供商（如Akamai）处理动态内容时，假设大多数CDN可以实现缓存，处理动态内容的服务器选择算法采用的是静态算法的一个直接扩展算法。例如，

Akamai 使用 EdgeSuite 来处理动态内容，它将每一个网页视为由不同类型的片段构成，片段的类型可以是静态、动态可缓存或动态不可缓存。这样，一个由 EdgeSuite 使能的重定向器可以选择最近的服务器来处理静态和动态可缓存的片段，而将动态不可缓存的片段甩给源服务器。然后，重定向器将所有片段装配起来返回给用户。但据估计，只有大约 60% 的动态片段是可缓存的[30]，如果一个网站（如电子商务或拍卖网站）的内容中不可缓存的动态内容占相当大的比例，那么用户可能会感觉到相当长的下载时间，因为从源服务器下载那些不可缓存的片段占用了大量的时间。因此，针对此类网站的内容分发策略可以从服务器端重定向方法的使用中受益。

6.3 背景

在本节中，首先介绍 Web 集群的多层架构，以及排队论的一个简要背景。

6.3.1 集群的架构

下面以电子商务会话中的请求为例，解释图 6.1 所示的 Web 集群的多层架构。首先，初始集群收到一个客户端的请求。初始集群的选择可以依据客户端重定向策略，如 DNS 轮转或参考文献 [17, 20] 中那些更复杂的策略。初始集群收到请求后，一个分派器使用轮转方式或其他复杂策略[14]将请求发送到 Web 层的一个服务器上。如果该请求的目标是一个静态网页，那么它就由一个 Web 服务器来处理，接着该服务器生成响应信息并将其返回给用户。如果请求的是动态内容（如订单处理或购物车），那么该请求被转发给应用层的一个服务器，由该服务器对客户端的请求进行分析，解析出所需的全部动态片段，并产生相应的数据库查询操作。至于由哪个数据库服务器处理查询的操作，则由另一个分派器决定。最后，应用服务器对数据库查询操作的所有响应信息进行整理，组装为页面返回给 Web 服务器，Web 服务器再将其转发给客户端。

6.3.2 排队论

解析模型将在后面的 6.5 节给出，它将一个服务器表示为一个 M/G/1 队列[29]。在一个 M/G/1 队列中，连续两个请求之间的到达间隔时间服从指数分布，对请求的服务时间服从一个任意的分布，其中，M 和 G 分别代表马尔科夫分布和一般分布（general distribution），1 表示只有一个队列。M/G/1 队列比 M/M/1 队列更一般化，M/M/1 队列中请求的到达间隔时间和服务时间都服从指数分布。不过，M/G/1 队列的一般性越高，则付出的成本就越高。在稳定状态下，一个 M/G/1 队列中的任务数不服从一个一般闭式（closed form）的分布，不过队列中任务数的平均值和平均处理时间的确存在一般解。

虽然本节用一个 M/G/1 队列表示一个服务器，但需要指出的是，G/G/1 队列应该是对一个服务器更一般也或许是更精确的表示。当假设到达间隔时间服从指数分布时，隐含的假设就是在当前时刻到达网站的请求与以往的请求是相互独立的。不过，

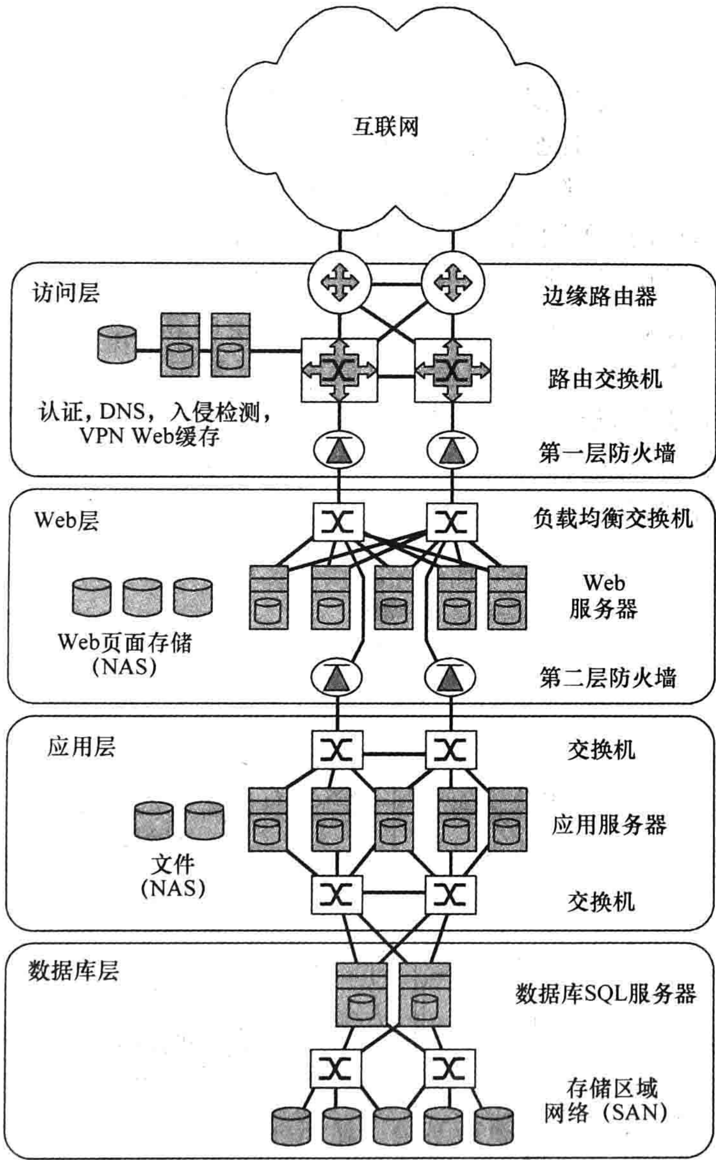


图 6.1　集群的多层架构

网站收到的请求之间确实存在一些相关性，例如属于同一个会话的那些请求之间是相关的。而且，后端层（如数据库层）的请求之间可以认为具有更大的相关性，因为一个 Web 请求可能分解为多个相关的数据库查询操作。但为了简化，这里假设一个 M/G/1 队列可以将队列平均等待时间表示为一个闭式的表达式，这就可以推导出可重定向请求所占的比例。

6.4　重定向架构和算法

本节介绍 WARD，即动态内容的广域重定向架构，并给出一个基于测量的重定向算法。

6.4.1 WARD

在 CDN 中，服务和应用在集群之间复制，这些集群之间使用高速连接。在实际中，一个操作员管理在地理上分布于不同地点的多个集群。首先，一个客户端的请求到达一个初始的集群。这里，初始集群的选择方法可以是静态或动态的，它们既可以使用简单的 DNS 轮转策略，也可以使用更复杂的考虑到内容可用性和网络延时的策略，其中网络延时是由代理服务器按一定的周期测量而得[17,20]。图 6.2 所示是一个分派器，它使用下面将要提到的重定向算法，隐含地将请求转发到一个远程的集群。重定向算法的目标是将请求进行重定向，但只有在重定向到远程集群后节省的请求处理时间超过请求双向穿行于网络主干中带来的网络延时，才进行重定向操作。在这种方式下，只对分派器进行修改而不改变其他部分就可以使端对端延时得到降低。

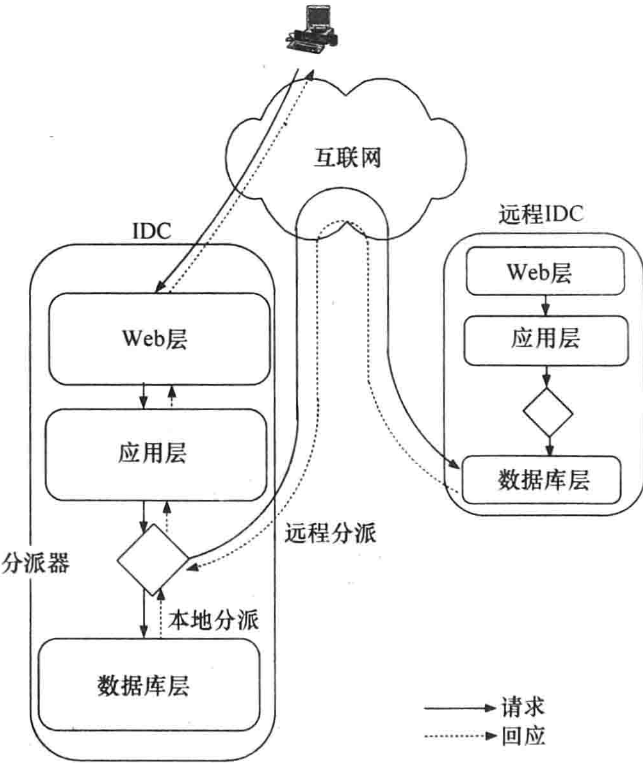


图 6.2 利用数据库层前置的重定向器实现广域重定向

因此，WARD 为集群资源的空间多路复用提供了基础。当某个集群由于瞬时拥塞[24,31]或每日时段效应[33]成为一个热点时，其负载可以被透明地重定向到其他的集群，以确保网络延时的降低。例如，当我们发现客户端访问模式呈现为每日时段效应，即服务器利用率随每天的不同时段变化时，那么就可以利用这个效应，使每个集群不必按照应付峰值的标准进行资源配置。这样，当一个集群的工作负载达到峰值时，距离它几个时区的另一个集群的负载将会低得多，采用集群间的重定向技术可以大大提高性能。

6.4.2　重定向算法

重定向算法的目标是实现请求处理时间的最小化，即如果一个请求到达集群 k，则将请求分派到满足以下条件的集群 j

$$\operatorname{argmin}_j(2\Delta_{kj} + T_j) \tag{6.1}$$

式中，Δ_{kj}表示集群 k 与 j 之间的网络延时；T_j为集群 j 对请求的处理时间。

在实际中，位于远处的集群 j 对请求的处理时间 T_j不可能事先知道，但可以通过集群 j 的平均负载以及请求类型进行估计。这样就可以使用一个基于测量的算法，其中平均处理时间 T_j用集群 j 的平均负载 ρ_j和请求类型估计。在 WARD 中，针对每种请求类型测量一个平均延时与负载的对应关系，以此实现对 T_j的估计。各个集群之间周期性地交换负载信息，实现对每个集群的处理延时的精确估计。与之不同的是，由于集群之间使用的是高速连接，因此集群之间的 Δ_{kj}相对保持不变。这样，在集群 j 收到请求后，分派器根据这条请求的类型所对应的延时—负载对照表，使用集群 j 负载的测量值估计集群 j 的总服务时间。

现在考虑另一个策略：它不是以单个请求为基础进行处理，而是计算被分派到远程的请求的比例。下一节将会提到，如果采取某些简化，则被分派到远程的请求与所有请求之间应该存在一个最优的比率，在这个比率下所有请求的总延时最小。当解算出这个比率后，分派器就可以简单地利用这个计算出的概率对请求进行重定向。这里将上述两种策略分别称为每请求（或每查询）重定向和概率重定向。

6.5　性能模型

本节讨论一个广域重定向的性能模型。对于给定的工作负载、网络延时及处理时间的均值和方差，可以推导出在延时最小的要求下分派器将请求重定向到远程集群的比例。此外，本节计算总响应时间的均值，包括服务和等待时间以及端到端网络延时。接着，进行一个系统性的性能分析，估计最优分派比率 α^*，并对请求在各种参数组合下（服务器负载、端对端网络延时和请求的平均处理时间）的平均响应时间进行预测。

图 6.3 显示的是 WARD 的系统模型。到达集群 i 的请求数可以建模为参数为 λ_i的

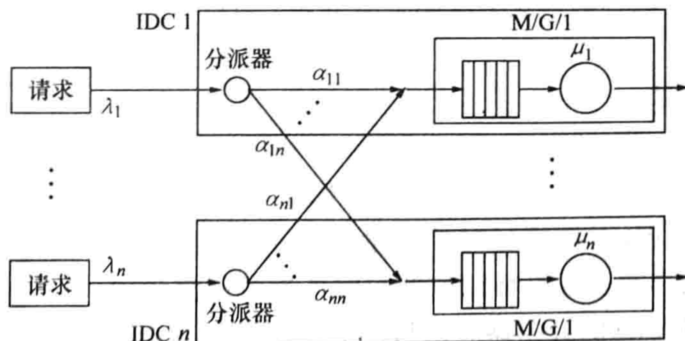


图 6.3　重定向系统模型

泊松过程。同时，使用服务时间的一般分布（general service time distribution）建模一个单瓶颈层，分布的均值为$\bar{x}_i$，方差为σ_i^2。

现在考虑一个概率重定向算法，它将一个请求以概率α_{ji}从集群j重定向到集群i。用$E[T_i]$表示集群i处理一个请求的平均总延时，Δ_{ji}表示请求从集群j发到集群i的端到端单程网络延时。

对于一个含有n个集群副本的系统，设$\boldsymbol{A}=\{\alpha_{11},\cdots,\alpha_{ji},\cdots,\alpha_{nn}\}$为请求分派比例矩阵，$\boldsymbol{E}[\boldsymbol{T}]=\{T_1,\cdots,T_n\}$为每个集群的瓶颈层总延时构成的向量，$\boldsymbol{D}=\{2\Delta_{11},\cdots,\Delta_{ji}+\Delta_{ij},\cdots,2\Delta_{nn}\}$表示数据从集群$i$到集群$j$的往返时间矩阵，$\boldsymbol{L}=\{\lambda_1,\cdots,\lambda_n\}$为在集群分派器处的请求到达率向量，$\overline{\boldsymbol{X}}=\{\bar{x}_1,\cdots,\bar{x}_n\}$为平均服务时间，$\boldsymbol{C}=\{c_1,\cdots,c_n\}$是服务时间的平方变异系数（squared coefficient of variation）向量，其中$c_i=\sigma_i^2/\bar{x}_i^2$。

引理：当请求分派比例为$\boldsymbol{A}$时，重定向策略的平均服务时间为

$$\boldsymbol{E}[\boldsymbol{T}]=\boldsymbol{A}\cdot\overline{\boldsymbol{X}}+\frac{(\boldsymbol{A}\cdot\boldsymbol{L})\overline{\boldsymbol{X}}^2(1+\boldsymbol{C}^2)}{2[1-(\boldsymbol{A}\cdot\boldsymbol{L})\overline{\boldsymbol{X}}]}+\boldsymbol{A}\cdot\boldsymbol{D} \tag{6.2}$$

在本章的附录部分将给出引理1的证明。该证明是基于M/G/1队列平均等待时间的闭式表达式，从式（6.2）可以计算出使所有请求的服务时间最小的最优分派率。设$\boldsymbol{A}=\{\alpha_{11}^*,\cdots,\alpha_{nn}^*\}$为最优请求分派率矩阵。

命题1：最优分派比率$\boldsymbol{A}^*$由式（6.3）给出。

$$\frac{\partial}{\partial\alpha}\left\{\boldsymbol{A}\cdot\overline{\boldsymbol{X}}+\frac{(\boldsymbol{A}\cdot\boldsymbol{L})\overline{\boldsymbol{X}}^2(1+\boldsymbol{C}^2)}{2[1-(\boldsymbol{A}\cdot\boldsymbol{L})\overline{\boldsymbol{X}}]}+\boldsymbol{A}\cdot\boldsymbol{D}\right\}=0 \tag{6.3}$$

式中，$\boldsymbol{E}[\boldsymbol{T}]$由式（6.2）定义。

为了求解式（6.3），使用下列约束来减少未知数的个数：首先，令$\sum_j\alpha_{ji}^*=1$；其次，$\lambda_i\geqslant\lambda_j\Rightarrow\alpha_{ji}^*=0$，即对于两个$\lambda$值不同的集群，在平稳状态下请求不会从到达率较低的集群分派到较高的集群。

最优分派率$\boldsymbol{A}^*$可以用于预测一个集群副本系统的平均请求服务时间。

命题2：最优分派率所对应的请求服务时间的均值由式（6.4）给出：

$$\boldsymbol{E}[\boldsymbol{T}^*]=\boldsymbol{A}^*\cdot\overline{\boldsymbol{X}}+\frac{(\boldsymbol{A}^*\cdot\boldsymbol{L})\overline{\boldsymbol{X}}^2(1+\boldsymbol{C}^2)}{2[1-(\boldsymbol{A}^*\cdot\boldsymbol{L})\overline{\boldsymbol{X}}]}+\boldsymbol{A}^*\cdot\boldsymbol{D} \tag{6.4}$$

证明：

使用式（6.3）得到的最优分派率，由引理1可以得到式（6.4）。

6.6 数值结果

下面的结果将显示，广域重定向可以降低客户端处的总访问延时。相比于服务器负载测量误差，WARD对网络延时的测量精度要求较低，这就进一步验证了这里的假设：重定向机制必须获得更精细粒度的服务器负载测量值，因此WARD更适用于以下情况：运行WARD的分派器应该与其他本地服务器同地协作（co-located），且通过高速连接与远程服务器通信。

使用6.5节给出的系统模型，考虑一个由两个集群构成的系统，其副本具有相同的平均请求服务时间 $\bar{x}$。进而，假设这是一个对称的网络，其中两个集群之间的广域延时 $\Delta=\Delta_{12}=\Delta_{21}$。最后，设 $\lambda_2=0$，即满足 $\lambda_1>\lambda_2 \Rightarrow \alpha_{21}=0$，为简单起见表示为：$\lambda := \lambda_1$、$\alpha^* := \alpha_{11}{}^*$。

基于式（6.2），分派率的计算公式为

$$E[T]=\alpha\bar{x}+\frac{\alpha\lambda\bar{x}^2(1+c^2)}{2(1-\alpha\lambda\bar{x})}+(1-\alpha)\bar{x}+\frac{(1-\alpha)\lambda\bar{x}^2(1+c^2)}{2[1-(1-\alpha)\lambda\bar{x}]}+2(1-\alpha)\Delta \quad (6.5)$$

根据命题1可以求解式（6.5），从而得到最优分派率 α^*。从现在开始，将 $1-\alpha^*$ 称为远程重定向率，即分派到远程处理的请求的比例。由命题2，用式（6.5）中最优比率 α^* 替换 α 可以求得这个集群系统的平均总延时。

如无特殊说明，本节使用以下默认值：$\bar{x}=42.9$ms、$\sigma=40.1$ms。这些值是在 Δ 设置为36ms时由测试平台计算得到的，Δ 相当于两个集群之间在纬度45°相隔6个时区的光速延时[⊖]。这里使用 $\rho=\lambda\bar{x}$ 表示所有集群的总负载。对于那些没有重定向的集群，ρ 相当于服务器在瓶颈层的负载，因此WARD可以将负载分派到本地集群和远程集群上。为了得到 ρ 的一个给定值，可以将 $\bar{x}$ 保持固定不变，而调节到达率 λ 的取值。

6.6.1　广域重定向下的结果

首先证明广域重定向可以降低用户感知到的总延时。使用式（6.3）和式（6.4）计算WARD的总延时，并将其与不采用重定向的集群的总延时进行比较。从图6.4可以看到，总延时是端对端延时和不同的系统负载 ρ 的函数。

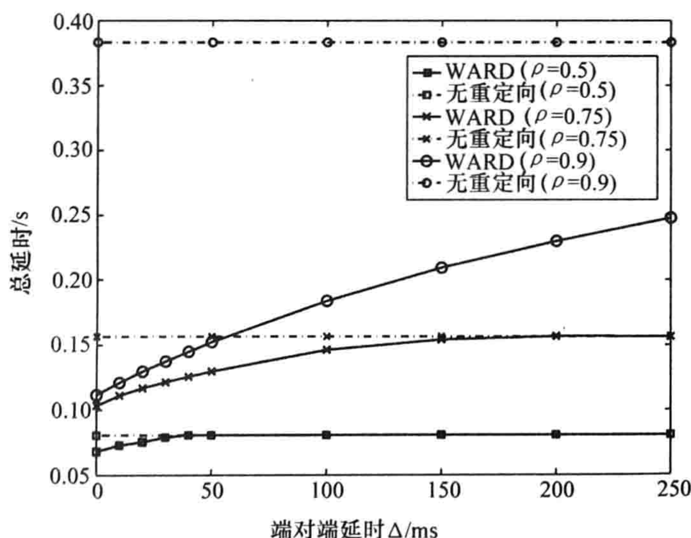


图6.4　使用和不使用广域重定向时总延时的比较

在低负载（$\rho=0.5$）的情况下，只有在端对端延时 $\Delta\leqslant 25$ms 时总负载才会有所降低；对于较高的延时，重定向开销超过了处理的时间，此时所有的请求将在本地处理。

⊖　设地球在纬度为45°处的圆周长度为28 335km，且光在光纤中的速度为205km/s，则横跨1个时区的单程延时为28 335/（205×24）≈6ms。——原书注

不过，当负载较高时总延时可以得到显著的降低。在一个中间水平的负载$\rho=0.75$以及$\Delta<50$ms 的情况下，使用 WARD 可以使总延时从 0.16s 降低到 0.13s，降低了 18%。对于一个负载很大的系统（$\rho=0.9$且$\Delta<50$ms），如果不采用重定向，总延时为 0.38s，而采用 WARD 时总延时可以降低到 0.15s，降低了 60%。当负载ρ超过 0.9 时，相信该模型能够更大幅度地降低延时。

6.6.2 对测量误差的敏感度

下面将证明只有在服务器利用率采用一个非常精细的测量粒度时才能获得性能上的收益。与之形成对照的是，网络延时的测量粒度即使较大也不会产生不利的影响。为了证明这一点，这里研究了网络延时Δ和服务器负载ρ的测量误差对性能的影响。在进行分析时，采用误差容忍度这个指标来量化对性能的影响。误差容忍度定义为对总延时的影响最大不超过 2% 的百分比误差$\pm\varepsilon$。

首先研究网络延时的测量误差对性能的影响。令Δ表示实际的集群间网络延时，$\hat{\Delta}=\Delta+\delta$表示其测量值，$\hat{\boldsymbol{D}}$表示相应的往返时间矩阵。分派器计算分派率时，用$\hat{\boldsymbol{D}}$替换式（6.2）中的$\boldsymbol{D}$。

利用式（6.4）计算总延时的平均值时，将使用实际的网络延时$\boldsymbol{D}$。网络延时的测量误差对远程重定向率（$1-\alpha$）及相应的平均请求响应时间的影响如图 6.5 所示。每条曲线表示一个不同的（真实的）延时Δ，x轴表示误差δ（为Δ的百分比）。

从图 6.5a 中可以看出，δ为负时，重定向率的变化幅度比相应的正值处要大。产生这种现象的原因是，重定向率并不随端对端延时而线性增长。非对称性的一个表现是，总延时在δ为负时增长的更多，如图 6.5b 所示。不过，需要注意的是，响应时间对延时测量误差并不是很敏感，误差容忍度在 ±20% 之间相当高。

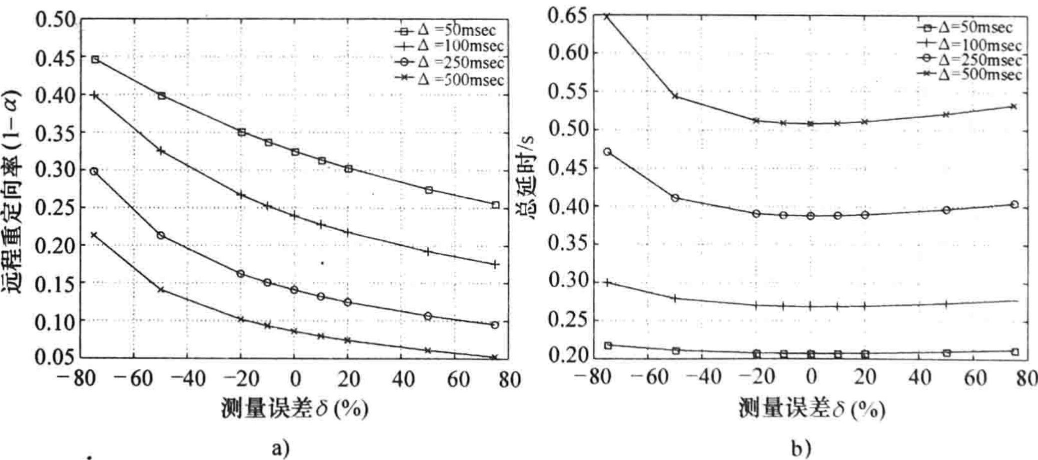


图 6.5 在网络测量误差下的 WARD 性能（x轴上的 0 点表示没有测量误差的理想情况，服务器负载设置为$\rho=0.95$）

a）远程重定向率 b）总延时

类似地，可以研究服务器负载测量不精确（如源自接收测量数据时的延时）的情况。这种情况下，分派器测量到的负载为$\hat{\rho}=\hat{\lambda}\bar{x}$，其中$\hat{\lambda}=\lambda+\varepsilon$（$\varepsilon$取值为正确负

载 ρ 的百分比），测量到的相应到达率为 $\hat{L}$，网络延时设置为 $\Delta = 500\text{ms}$。

首先分析测量误差 $\varepsilon > 0$ 的情况。这时，分派器假定服务器负载比其实际值高，从而进行重定向的请求比最优值要多。从图 6.6a 可以发现，远程重定向率随测量误差的增大而增大。这些额外的重定向带来了更多的网络延时，且总延时随之线性增加，如图 6.6b 所示。特别地，当 $\rho \geqslant 0.9$ 时，误差的容忍度为 +1.5%。

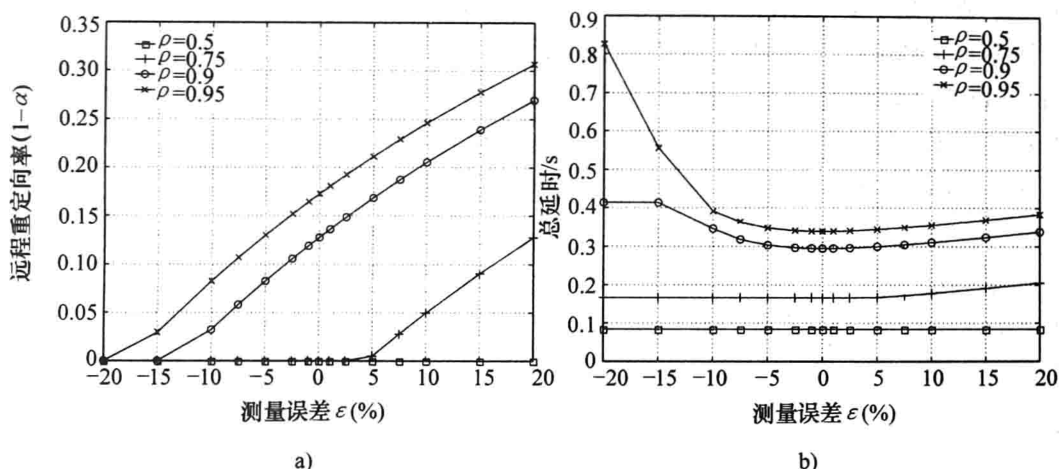


图 6.6　服务器负载测量误差与 WARD 性能的关系

a）远程重定向率　b）总延时

当 ε 为负时，分派器假定本地服务器负载小于实际值。这时，重定向数量减少，本地服务器的负载给本地集群带来了更长的处理时间。正如预期的那样，图 6.6b 显示，在服务器运行于高负载时（$\rho \geqslant 0.9$），总延时对负的 ε 敏感得多，这时误差容忍度只有 -0.5%。

对延时测量误差和服务器负载测量误差造成的影响进行比较可知，对延时误差的误差容忍度为 ±20%，而对服务器负载的误差容忍度则低一个数量级，为 +1.5% 和 -0.5%。从中可以得出结论：服务器负载比网络延时需要更高的测量精度。

对于服务器端重定向机制来说，由于它能够在较低的开销下得到更精确的服务器负载信息，所以自然可以更有效地实现请求的均衡。首先，运行在客户端或 DNS 服务器的客户端重定向策略也许不能使用服务器的高带宽连接；其次，运行在接近客户端的代理服务器（如 Akamai）上的客户端重定向也许能使用高带宽的访问连接，但客户端需要配备比服务器端更多的分派器，这使得获取服务器负载信息的开销要大得多。设集群的数量为 n，即 WARD 中有 n 个服务器端分派器，如果在运行 WARD 的那个层中，所有集群的服务器总数为 $M(M > n)$，那么 WARD 中交换信息的复杂度是 $O(nM)$。与之相比，设每个客户端附近的代理服务器具有一个分派器，那么总共有 N 个代理服务器或分派器的客户端重定向机制的交换信息复杂度为 $O(NM)$。假定在典型情况下客户端代理服务器数量远大于集群副本的数量，则服务器端重定向机制的复杂度要小得多。这一点可以通过下面的例子证明：假设在一个客户端重定向方法中，每个自治系统中安装一个代理服务器，则 N 的数量估计为 39 000 左右，相比而言，一个典型的集群网格包含大约 60 个集群，而 $60 \ll 39\,000$。

6.7 测试平台的实现及实验

本节介绍一个 CDN 原型的实现，通过实验将使用 WARD 的广域重定向与其他的重定向策略进行比较。实验结果为广域集群驱动的重定向和平台关键性能因子实验提供了一个概念验证的展示，并验证这个性能模型。

如图 6.1 所示，测试平台由一个集群构成，该集群中每台计算机的配置为：Intel Pentium Ⅳ 2.0 GHz 处理器、操作系统为 Linux 2.4.18 - 14、512 MB SDRAM 内存和一个 30GB ATA - 66 硬盘。集群中的一台计算机作为一个路由器，运行 Nistnet[6]（即一个 IPlayer 网络仿真软件包），这台路由器将其他计算机分为三个域：客户端和两个集群。这一配置可以生成客户端和集群之间及集群和集群之间的网络条件（延时和带宽）的不同组合。这里提出的一个三层系统架构如图 6.2 所示，其中 Web 层使用了一个 Apache Web 服务器[8]，在应用层中动态内容使用 PHP 脚本提供[7]，使用一个 MySQL 服务器实现对一个容量为 4GB 的数据库进行访问[5]。

6.7.1 数据库分派器

上述集群架构的一个关键部分是分派器，它负责决定是否对一个本地或远程的请求进行服务。在这里的测试平台中，由于复杂的数据库操作所带来的大量处理需要，使数据库层在最初阶段就成了瓶颈，因此在数据库前端配置了分派器用于远程重定向。在我们的实现中，提供了 Web 层和应用层，并为其配备充分的资源，以便使数据库层确实成为一个瓶颈。

因为数据库分派器的目标是使查询的响应时间最小化，所以它的分派能力受到一致性约束的限制。分派器使用两种方式解决一致性问题：一是对所有的数据库服务器保持写操作的统一有序；二是“一读全写”的分派策略。为了保持对所有写操作统一有序，每一个写操作被分配一个唯一的序列号，数据库服务器按照序列号的顺序处理写操作。在“一读全写”分派策略中，写操作被发送到所有的数据库，一旦其中有一个服务器完成了该操作，响应就返回，因此数据库一致性的维护是异步的。如果是一个读操作，则将其有选择地发到一台数据库服务器上，服务器的选择依据是，该服务器上相应的冲突写操作已通过一个称为冲突检测[11]的调度策略进行了处理。这种异步维护的一致性和冲突检测机制相对于同步一致性算法，表现出了更高的吞吐量。

不过，虽然冲突检测调度使这组服务器对读操作不能进行大范围内的分派，但两个因素的存在使我们可以忽略这个局限：一是对读操作的服务时间长于写操作；二是电子商务中读的比重非常大[10]。

WARD 允许使用一个一般的框架在任意层实现远程重定向。虽然这里将其实现在数据库层的前端，但它也可以完全等效地实现在 Web 层或应用层的前端，这时可以重定向整个请求（而不是一组数据库查询）。请求重定向可以降低网络开销，但它同时也会将数据库操作限制到本地数据库服务器，因此就不能获得将数据库操作进行远程重定向所带来的好处。本章不对请求重定向和数据库操作重定向进行对比，而是

将注意力集中在一般意义上远程重定向对性能产生的提高。

6.7.2　重定向算法

在一致性约束下，数据库分派器将读操作定向到具有最小平均响应时间的数据库服务器上。首先，在所有的集群中使用冲突检测调度，那些未处理写冲突的集群被排除。然后，在剩下的集群中，分派器使用每请求或概率重定向策略选择一个集群。在每请求策略中，分派器使用数据库层负载的测量值计算平均响应时间，并确定服务器处理时间的节省是否大于分派导致的延时。在概率策略中，分派器使用根据模型计算出的最优重定向率，并按照此概率分派请求。通过实现这个概率策略，使得对于给定的若干客户端，使用模型预测的重定向率进行配置，就可以不必在线测量服务器的负载。

6.7.3　TPC－W 工作负载

对于工作负载，实验中采用 TPC－W 基准[9]生成一个电子商务的工作负载，用其模拟一个在线书店网站。这里使用由 Amza 等人开发的针对冲突检测调度和动态内容应用的策略[11]。

TPC－W 的工作负载用一个客户端仿真器产生。仿真器生成 TPC－W 浏览操作集合中规定的请求，包含 95% 的读操作和 5% 的写操作。这个客户端仿真器启动 n 个用户会话，延续时间为 15min。每一个会话与 Web 服务器建立一个永久的 HTTP 连接，并发送一个请求序列到集群。在两个请求之间，用户在发出下一个请求之前的等待时间是一个可配置的参数，称为思考时间（think time）。思考时间的平均值（这里设为 7s）和用户数可以用来定义集群的请求到达率。这里需要注意的是，一个 PHP 脚本可能包含很多个对数据库的操作。操作的提取和序列化执行由应用层执行。由于每个请求包含了若干个内嵌的数据库操作，它们到达数据库的时间分布和到达率与由客户端仿真器产生的情形是不同的。

6.7.4　实验

在实验中，输入参数包括集群间连接延时（通过 Nistnet 模块来调整）和到达每个集群的客户端数量。测量的参数包括客户端测量到的请求响应时间、数据库分派器测量到的数据库请求响应时间和远程重定向率。请求响应时间定义为一个请求从产生到客户端收到响应的最后一个字节之间的时间。数据库请求响应时间定义为分派器向数据库发出一个请求到分派器接收到响应之间的时间。对于实验过程中产生的所有请求，计算响应时间的均值及其第 90 百分位数。远程重定向率定义为分派器转发到远程数据库服务器的请求数占总请求数的比例。

首先，使用离线技术根据数据库请求响应时间的特性对每请求重定向策略进行配置；其次，在本地数据库服务器的负载与广域连接延时之间寻求一个折中，从而量化 WARD 架构在性能上的收益；第三，根据测试平台获得的测量值，使用 6.5 节中的解析模型计算性能的收益，并与实际情况进行比较。

1. 查询响应时间特性的离线测量

在实验中，将响应时间作为 CPU 负载的一个函数进行测量，对于每请求重定向策略来说这是一个关键性的输入条件。实验中使用的每个集群可以访问一个本地数据库服务器。一个数据库请求的执行时间取决于同时运行在数据库服务器上的其他请求的数量和类别，这可以被抽象为进入系统的工作负载，从而可以通过增加客户端的数量来改变数据库服务器的 CPU 负载。在每个情况下，对每组（30 个）MySQL 只读请求测量其平均执行时间，得到的延时—负载关系曲线（见图 6.7）将被用于每请求重定向策略。

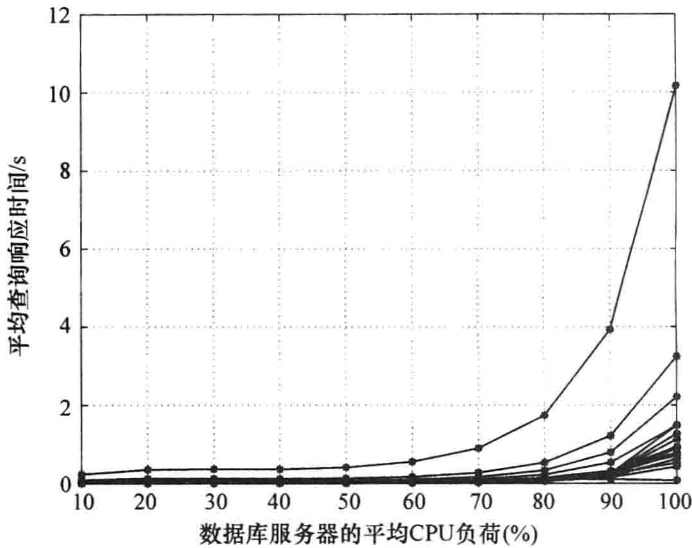


图 6.7 平均查询响应时间和数据库服务器的平均 CPU 负载
（使用 TPC - W 基准的浏览集中 30 个只读的 MySQL 请求）

2. WARD 性能收益

WARD 可以结合服务器负载和网络延时执行一个每请求重定向。利用这里的实验测试平台，首先验证 WARD 在改善集群性能方面的功效。实验中，为本地集群的 Web 层配备充足的服务器，使其不再成为瓶颈。本地和远程集群的数据库层都有一个服务器，请求只到达本地集群，在其 Web 层处理完后，其中对数据库的请求或者在本地数据库层处理，或者重定向到远程的数据库层。

这里将 WARD 的性能与下述 4 种不同的 strawman 算法进行比较：①无重定向，即本地处理所有的查询；②只考虑延时，即根据最近一次测量的结果，将请求转发到往返时间最小的那台服务器上；③只考虑服务器，即数据库请求被转发到 CPU 负载最小、被认为处理最快的服务器上；④轮转，即使用轮转方式在本地和远程数据库服务器之间转发请求。后三种涉及数据库请求整体重定向的算法被称为重定向 strawman 算法。

实验中，TPC - W 包含 100 个客户端会话，每个会话的平均请求到达间隔时间为 4s，得到的性能如图 6.8 所示。首先，从图 6.8a 中可以看出，所有将请求从本地服务器重定向到远程服务器的算法，其性能都优于无重定向的算法。这就重申了以下事实：对于这种程度的工作负载，本地数据库服务器负担很重，所以将其一部分负载重定向到远程服务器会取得更好的效果。不过，WARD 的性能比所有的 strawman 算法

都要好得多，这就证明了其优越性。实际上，如图6.8b所示，WARD的性能较高，因而数据库请求被重定向到远程服务器的比例就最低。产生这个现象的原因在于WARD采用了更好的基于每请求策略的重定向方法，而每请求策略的优势是综合考虑了服务器负载和延时这两方面的因素。

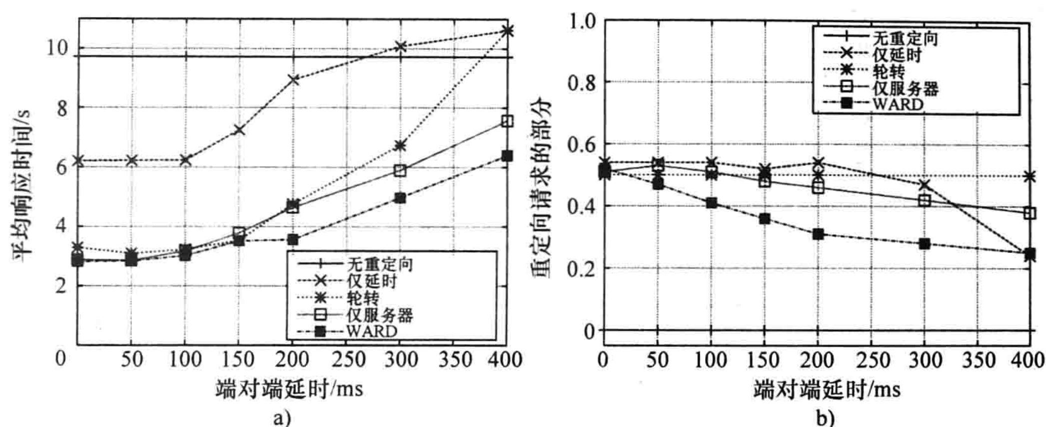


图6.8　WARD与无重定向、仅延时、仅服务器和轮转等strawman算法的性能比较

a）响应时间　b）重定向率

其次，只考虑延时的算法在所有的strawman重定向算法中性能最差。这个结果进一步验证了这里的假设：对于一个动态内容网站（如实验中的这个模拟环境），在进行转发决策时服务器负载的影响比延时更重要。不过，当集群间延时增大时，只考虑服务器负载的算法在性能上也变得比WARD更差了。这也是一个意料中的结果，因为只考虑服务器负载的算法并不将延时因素结合到其重定向决策中，因此就不能为每个请求做出正确的重定向决策。相反，WARD在进行重定向决策时，从设计上就综合考虑到服务器负载和延时。

第三，所有重定向算法（WARD和strawman算法）的性能都会随着集群间延时的增大而下降，其原因在于重定向的延时开销变得更高了。最终，即使单条数据库请求的重定向开销也可能会超过一台低CPU负载的远程服务器的收益。因此，一旦端对端延时高达270（375）ms，只考虑延时的算法（轮转算法）甚至比无重定向算法还要差。相比所有的strawman算法而言，WARD可以支持更远的远程集群。

3. 模型的验证和重定向策略

为了验证6.5节提出的解析模型，在这里的测试平台上将其与WARD重定向策略进行了比较。因为瓶颈在数据库层，所以这里将该模型下数据库层处理请求的重定向率和响应时间与测试平台进行了比较。对于该模型，这里使用了6.6节中的式(6.5)，参数设置为$\bar{x}=42.9$ms、$\sigma=40.1$ms，这与测试平台上在未加载的数据库服务器上测量到的结果相同。

首先，图6.9中将模型与单集群中实际的平均请求响应时间（服务器负载ρ的函数）进行了比较。从中可以发现，当服务器负载$\rho<0.7$时，模型与实际测量到的响应时间的符合度差异在±10%以内。服务器负载ρ超过0.7时，模型的结果与实际结

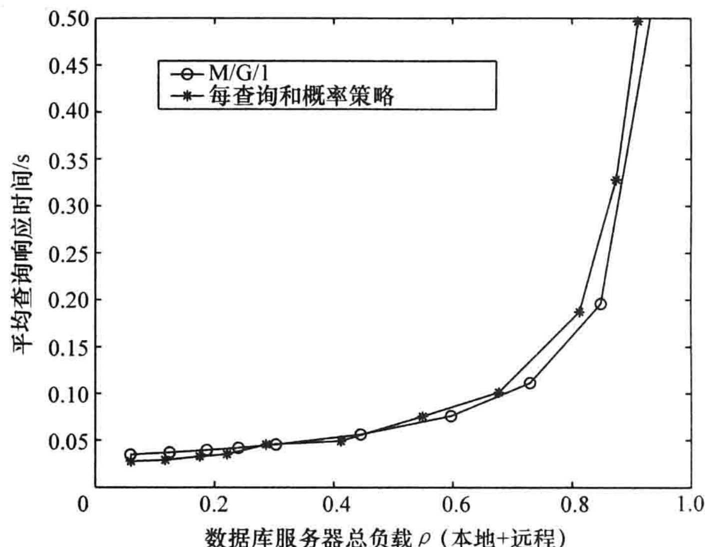


图 6.9 WARD 解析模型（单集群）

果存在较大差异，这是由于：①这里的 M/G/1 模型是基于一个简单的假设，即服务器层每个请求的响应返回过程相互独立，这个假设可能与实际情况相悖，例如由同一条 Web 请求所派生出的若干数据库请求之间存在着相关性；②这里的 M/G/1 模型没有考虑到读—写冲突，当存在冲突时数据库请求的处理时间可能会比模型预计的时间长；③在高负载情况下，会存在更多的数据库请求，并由此出现更多的冲突。

其次，对模型实现重定向的概率策略和每请求策略进行了比较。在所有实验中集群间延时设为 25ms，采用每请求策略时每隔 5s 接收一次 CPU 负载的测量值。

在图 6.10 中比较了远程重定向率和请求响应时间与系统负载的关系。模型的重定向率与概率策略关系紧密，因为该策略基于模型预测的最优值。另外，相比于这里的模型和概率策略，每请求策略在 $\rho<0.5$ 时可以更早地启动重定向，从而被重定向的请求也更多。出现这种情况的原因在于，请求量很大时会对系统负载比较敏感，如图 6.10a 所示。由此，有必要在系统负载相对较低的时候就启动请求的重定向。因此，每请求策略在 $\rho<0.5$ 时表现出较好的性能，平均响应时间较低，如图 6.10b 所示。

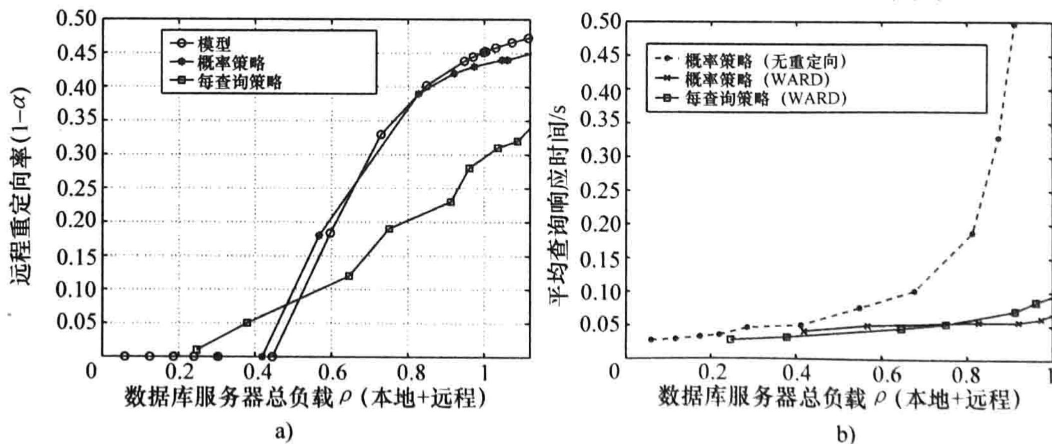


图 6.10 WARD 解析模型的验证

当$\rho>0.5$时，概率策略重定向请求的数量比每请求策略多，也就产生了更低的响应时间。这种差异的原因是，5s的采样间隔造成时间粒度过粗，以至于不能捕获CPU负载的小波动。采样间隔小一点应该可以得到更好的响应时间，但这就要求在精度和测量开销上寻求一个最优的折中。

这样，从实验中得出了两个重要的结论：第一，尽管M/G/1模型非常简化，但它的确与实际情况匹配得很好，所以6.6节中的模型得出的结论应该可以认为与真实世界的情况相吻合；第二，WARD的每请求重定向算法与概率算法相比，在低负载的条件下效果更好，而在高负载情况下可以通过减小采样间隔来改善其性能。

6.8 写给使用者

首先，本章强调了请求重定向对CDN资源多路复用和Web用户响应时间的重要性。基于此，本章研究了一个被世界各地用户访问的Web应用。资料显示，Web应用具有“每日时段”的流量特点，即白天工作时间的流量比晚上高。假设客户端重定向机制的主要目标是将用户重定向到最“近”的服务器集群，那么可以认为某个特定的集群在白天时间具有较高的流量。如果由CDN管理的一个应用使用了跨越欧洲和北美的两个集群，那么由于它们之间时间的差异，多路复用可以明显实现资源的节省。如果流量在美国的白天超过了正常水平，一个广域重定向机制能够将请求从过载的集群（位于美国）重定向到暂时未过载的集群（位于欧洲），所以广域重定向机制可以做到资源的多路并行复用。既然位于美国和欧洲的集群可以利用这个机制获得类似的性能，那就都不需要过多地配备其资源，从而实现资源的节省。

第二个结论是提供每请求重定向或服务器选择机制的重要性，这些机制可以将服务器负载和网络延时综合考虑。基于此，在6.6.2节中强调了服务器端重定向机制能够在较低的负载下以精细粒度测量服务器的负载。这就重申了一个重要的发现：相比于客户端重定向，一个服务器端重定向机制可以在较低的测量开销下取得较高的性能（如Akamai中的DNS重定向）。

6.9 未来研究方向

本章探讨了通过使用一个请求重定向机制获得的性能提升，并为动态内容设计了一个请求重定向算法。因为数据库层被认为是瓶颈所在，所以将这个重定向算法在数据库层进行了验证。

将来一个值得关注的研究方向可能是将重定向机制应用到集群的其他几个层（如Web层或应用层），以期获得性能的进一步提高。在处理一个客户端请求时，应用层向数据库层发出若干条数据库请求，数据库层执行的重定向操作把所有这些数据库请求分派到位于一个远程集群的数据库服务器上。在这种情况下，每一个数据库请求都要承受网络的往返延时，而如果将该客户端请求（而不是那些派生出的数据库请求）分派到一个远程集群的Web服务器上，这些延时就可以避免了。

6.10 结论

本章提出的请求重定向技术，不仅可以降低用户端响应时间，还可以从整体上实现集群之间资源的多路复用。特别地，本章为动态内容开发了一个概念验证性的请求重定向机制（即 WARD），用来强调相应的设计原则。WARD 的目标是通过综合考虑网络和服务器的延时来使处理动态内容请求时端到端的延时最小化。本章设计了一个解析模型以及用于概念验证的具体实现，以展示平均请求响应时间的显著降低。例如，对于托管一个电子商务网站并处理300个并发用户的 CDN，WARD 可以将平均响应时间从 5s 缩短到 2.3s，降幅达 54%。不仅如此，模型预测：当 Web 请求中动态内容处理的复杂性增大时性能也会进一步提高。WARD 尤其适合于避免由于短期瓶颈（如瞬时拥塞）造成的响应时间的增大。如果重定向的延时开销不是过高，WARD 也可以利用大时间尺度下的趋势（如由每日时段效应造成的工作负载变化），由此就可以避免对 Web 集群提供昂贵的过度配置。最后，WARD 是一个服务器端的重定向，这个解决方案与内容复制、客户端重定向和服务器迁移策略互不影响，这就使得它可以与这些方法实现无缝的集成。

附录

这里提供引理 1 的一个证明。

证明：

总服务时间由三部分组成：

1）请求在分派器和远程集群之间传输的网络延时；

2）在集群上的排队时间；

3）集群的服务时间。

根据对称性，在下面的公式中将时间开销归于接收请求的集群 i。首先，假设分派器与本地集群之间的网络延时 $\Delta_{ii}=0$，那么网络延时就只产生在请求转发给远程集群的过程中：

$$\alpha_{ji}(\Delta_{ji}+\Delta_{ij}) \tag{6.6}$$

其次，考虑请求在集群队列中等待处理的平均等待时间。通常一个 M/G/1 队列的等待时间为

$$\frac{\rho\bar{x}(1+c^2)}{2(1-\rho)} \tag{6.7}$$

式中，$\rho=\lambda\bar{x}$。

对于任何一个集群 i，到达率 λ 是从所有的集群 j 分派到集群 i 的请求的和，即 $\lambda_i=\sum_j\alpha_{ji}\lambda_j$。使用该 λ，式（6.7）可以针对单个集群 i 改写为

$$\frac{(\sum_j\alpha_{ji}\lambda_j)\bar{x}_i^2(1+c_i^2)}{2[1-(\sum_j\alpha_{ji}\lambda_j)\bar{x}_i]} \tag{6.8}$$

最终，集群 i 对一个请求的服务时间为

$$\alpha_{ji}\bar{x}_i \tag{6.9}$$

对于一组集群，这三项的和可得式（6.2）。

致谢

本章中的一些材料出自 IEEE INFOCOM'04[34]。作者感谢 Edward Knightly 教授和 Roger Karrer 博士对本章中大多数概念所做出的精辟论述和建议。

参考文献

[1] Akamai Whitepaper: Turbo-charging Dynamic Web Sites with Akamai EdgeSuite. http://www.akamai.com/dl/whitepapers, 2000
[2] Akamai. http://www.akamai.com, 2007
[3] Limelight Networks. http://www.limelightnetworks.com, 2007
[4] Mirror Image Internet. http://www.mirror-image.com, 2007
[5] MySQL Database Server. http://www.mysql.com, 2007
[6] NISTNET: Network Emulation Package. http://snad.ncsl.nist.gov/itg/nistnet/, 2007
[7] PHP Scripting Language. http://www.php.net, 2007
[8] The Apache Software Foundation. http://www.apache.org, 2007
[9] TPC-W: Transaction Processing Council. http://www.tpc.org, 2007
[10] Amza, C., Cecchet, E., Chanda, A., Cox, A., Elnikety, S., Gil, R., Marguerite, J., Rajamani, K., Zwaenepoel, W. Specification and implementation of dynamic content benchmarks. In: IEEE Workshop on Workload Characterization (WWC-5), Austin, TX (2002)
[11] Amza, C., Cox, A., Zwaenepoel, W. Conflict-aware scheduling for dynamic content applications. In: USENIX Symposium on Internet Technologies and Systems, Seattle,WA (2003)
[12] Arlitt, M., Williamson, C. Internet web servers: Workload characterization and performance implications. IEEE/ACM Trans on Networking 5(5) (1997)
[13] Arlitt, M., Krishnamurthy, D., Rolia, J. Characterizing the scalability of a large web-based shopping system. ACM Trans on Internet Technology 1(1) (2001)
[14] Aron, M., Sanders, D., Druschel, P., Zwaenepoel, W. Scalable content-aware request distribution in cluster-based network servers. In: USENIX ATC (2000)
[15] Bouchenak, S., Mittal, S., Zwaenepoel, W. Using code transformation for consistent and transparent caching of dynamic web content. Tech. Rep. 200383, EPFL, Lausanne (2003)
[16] Cardellini, V., Colajanni, M., Yu, PS. Geographic load balancing for scalable distributed web systems. In: MASCOTS, San Francisco, CA (2000)
[17] Carter, R., Crovella, M. Server selection using dynamic path characterization in wide-area networks. In: IEEE INFOCOM, Kobe, Japan (1997)
[18] Chen, Y., Katz, R., Kubiatowicz, J. Dynamic replica placement for scalable content delivery. In: International Workshop on Peer to Peer Systems, Cambridge, MA (2002)
[19] Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., Welhl, B. Globally distributed content delivery. IEEE Internet Computing (2002)
[20] Fei, Z., Bhattacharjee, S., Zegura, E., Ammar, M. A novel server selection technique for improving the response time of a replicated service. In: IEEE INFOCOM, San Francisco, CA (1998)
[21] Fraleigh, C., Moon, S., Lyles, B., Cotton, C., Khan, M., Moll, D., Rockell, R., Seely, T., Diot, C. Packet-level Traffic Measurement from the Sprint IP Backbone. IEEE Network Magazine (2003)
[22] Guyton, J., Schwartz, M. Locating nearby copies of replicated internet servers. In: ACM SIGCOMM, Cambridge, MA (1995)

[23] Jamin, S., Jin, C., Kurc, A., Raz, D., Shavitt, Y. Constrained mirror placement on the internet. In: IEEE, INFOCOM., Anchorage, AK (2001)

[24] Jung, J., Krishnamurthy, B., Rabinovich, M. Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In: International World Wide Web Conference (2002)

[25] Kangasharju, J., Ross, K., Roberts, J. Performance evaluation of redirection schemes in content distribution networks. Computer Communications 24(2) (2001)

[26] Karger, D., Lehman, E., Leighton, T., Levine, M., Lewin, D., Panigrahy, R. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In: ACM Symposium on Theory of Computing (1997)

[27] Karger, D., Sherman, A., Berkhemier, A., Bogstad, B., Dhanidina, R., Iwamoto K., Kim, B., Matkins, L., Yerushalmi, Y. Web caching with consistent hashing. In: World Wide Web Conference (1999)

[28] Karlsson, M., Mahalingam, M. Do we need replica placement algorithms in content delivery networks. In: 7th International Workshop on Web Content Caching and Distribution (WCW) (2002)

[29] Kleinrock, L. "Queueing Systems, Volume II: Computer Applications". Wiley (1976)

[30] Myers, A., Chuang, J., Hengartner, U., Xie, Y., Zhuang, W., Zhang, H. A secure, publisher-centric web caching infrastructure. In: Proc. of IEEE INFOCOM (2001)

[31] Padmanabhan, VN., Sripanidkulchai, K. The case for cooperative networking. In: Intl. Workshop on Peer-to-Peer Systems, Cambridge, MA (2002)

[32] Qiu, L., Padmanabhan, V., Voelker, G. On the placement of web server replicas. In: IEEE INFOCOM, Anchorage, AK (2001)

[33] Ranjan, S., Rolia, J., Fu, H., Knightly, E. Qos-driven server migration for internet data centers. In: Intl. Workshop on Quality-of-Service, Miami, FL (2002)

[34] Ranjan, S., Karrer, R., Knightly, E. Wide area redirection of dynamic content in internet data centers. In: IEEE INFOCOM, HongKong (2004)

[35] Shaikh, A., Tewari, R., Agrawal, M. On the effectiveness of dns-based server selection. In: IEEE INFOCOM, Anchorage, AK (2001)

[36] Villela, D., Rubenstein, D. Performance analysis of server sharing collectives for content distribution. In: Intl. Workshop on Quality-of-Service (2003)

[37] Wang, L., Pai, V., Peterson, L. The effectiveness of request redirection on cdn robustness. In: OSDI, Boston, MA (2002)

第2部分　CDN建模与性能

第7章　CDN的经济学设计

Nicolas Christin，John Chuang 和 Jens Grossklags

7.1　引言

在过去的10年，互联网内容及其分发手段都发生了巨大的变化。在20世纪90年代中期，电子商务和Web服务的发展使HTTP信息成为互联网上的主要内容，这就使网络上的计算机能够被简单地区分为客户端或服务器。然而，随后的两个重要事件从根本上改变了互联网的面貌。

首先，1999年出现的Napster服务[1]宣告了对等计算时代的来临。P2P计算明显不同于之前的（目前是共存的）客户端/服务器模式，因为网络边缘处的任何一台主机都能作为信息的提供者。其次，高分辨率相机、摄像机或具有采样功能的高质量声卡的普及使得数字产品变得大众化。这些反过来导致了用户自己生成的内容的大量出现，尤其是在各种信息传播平台（如博客）越来越易于使用时，这种情况变得更加明显。

简而言之，终端用户目前已经具备了相应的工具和资源，使他们可以作为主动的内容发布者。不过，与此同时（也许是被这种情况推动?），也出现了一些新的现象，如公共地悲剧（tragedy of the commons）[23]。

一个典型的例子可以很好地解释公共地悲剧问题。假设有一个牧场，其中供养着固定数量的羊，这个牧场是一个被所有牧羊人共享的公共资源。每个牧羊人在他的羊群数量稍微增加时可以得到一个明显的收益：更多的羊意味着更多的羊毛，也就意味着更多的收入；而且，如果牧场足够大，自然没人会介意多一两只羊吃草。问题是，所有的牧羊人可能都会这么想，最终造成在牧场中的羊数大量增加。在最坏情况下（所谓的“悲剧”），牧场中的草迅速耗尽，羊全都饿死，最终所有的牧羊人都失去了他们的一切资源。公共地悲剧从社会视角说明了每个个体自身的动机（incentive）与其愿景之间的失调。

公共地悲剧现象已经出现在现代信息网络，如P2P系统中就有很多这种动机失调的例子。在一个目前被广泛引用的研究中[2]，Adar和Huberman发现Gnutella文件共享网络中多达70%的用户没有共享出任何数据，而是在“蹭车”，也就是只从服务

中收益而不做任何贡献。这项研究中发现，Gnutella 网络实质上是在靠 1% 的用户支撑着，这些用户对 50% 的请求进行服务。换句话说，这种“蹭车”的现象使得 Gnutella 网络实际上非常类似一个中央服务器式的集中型网络，这就与 P2P 网络的基本原则背道而驰——P2P 网络本应能够推动所有参与者之间的信息共享和合作。

在其他的相关研究中，也揭示了 P2P 网络中另外一些严重的蹭车现象[45]以及 ad-hoc 网络中的不公问题[25]。这些观察到的现象表明，用户基本上是理性的，绝大多数用户做事首先是基于自己的利益。这就是说，他们感兴趣的是能否从网络中获得最大的收益，并最小化其成本（即所谓“自私”行为）。

用户的这种理性（或自私性）对参与开发 CDN 的相关各方有什么影响呢？毕竟，大多数 CDN 不属于同一个单位，也就无法有力地控制所有的参与者和解决任何动机失调的问题。

实际上，对于 CDN 设计者来说，对参与者动机的研究与几个利益层面有关：

1）一个内容提供商可能需要将终端用户变为内容分发机制的协助者，以便提高对用户的服务质量和降低网络设施的成本[26]，这可能是一个很有吸引力的机制，尤其是对于大规模、带宽密集型内容（如视频点播）的提供商。以知名度不断上升的 YouTube[9]为例，对这类服务的需求在不久的将来会变得相当大，因此可能需要重新考虑分发设施的设计。

2）不同服务提供商经营的 CDN 可能需要互联，但由于服务提供商之间一般存在竞争关系，所以研究它们各自的动机可以有助于设计可行的服务级协议。

3）即使假设参与者都服从指挥（就像所有 CDN 设施都由一个单位控制和所有那样），这时一个重要的问题就是：为了保证系统正常运转，必须能够确定每个参与者需要为系统贡献的资源数量。这个分析允许网络构造者洞悉潜在的“热点”，即网络上那些处理负载极大使得投资远远超出其可用资源的部分。既然网络在巨大负载下工作时热点可能是主要的失效点，那么其辨识就变得至关重要。

4）理解网络参与者各方的成本和收益可以有助于选择切实可行的用户定价方案，这将在其他章节讨论。

所有这些问题可以引导我们从根本上理解在设计 CDN 结构时参与者的动机以及参与各方提供的资源。

虽然与网络技术相比这类研究还较新，但在应用数学和经济学领域已研究了很多年，一个典型的研究内容是博弈论，研究非协作实体之间的策略交互，一直以来用于理解和建模竞争市场，如证券市场中相互竞争的投资者。近年来的研究工作显示，当存在着（潜在的）自私参与者时，博弈论可以很好地为网络设计和评估提供一个规范背景[46]。

本章建立在博弈论研究的基础上，做出的贡献如下：首先，为一个内容分发覆盖网中的代理设计了一个新的代价模型，用于对 CDN 中动机失调的根本原因进行了分析。另外，研究了 P2P CDN 文献中提供的一些网络结构，以确定它们是否是激励相容的。最后，回顾了一些传统的博弈论假设，研究它们在 CDN 设计上的适用性。

本章余下内容组织如下：在 7.2 节，对分析中用到的博弈论概念进行简要的介绍，并回顾一些与本章讨论相关的研究；7.3 节给出代价模型；在 7.4 节，使用该模型对 CDN 连接建立中的动机进行了分析；7.5 节中，将本章对动机的分析与近年出现的网络设施相结合；7.6 节中使用数值仿真对上述分析进行验证；7.7 节讨论了分析中使用的一些假设条件在放宽后如何对结果产生影响；7.8 节中指出未来的研究方向；7.9 节中给出了总结和简要的结论。

7.2　背景和相关工作

如上所述，通过对用户理性行为的观察，系统构建者的兴趣越来越多地集中在将网络参与者视为自私[37]或竞争[46]的实体。这些概念已经应用于一些现有的 P2P 系统。例如，为了减少蹭车现象，一些知名的 P2P 系统（如 KaZaA 或 BitTorrent[16]）采用了一些简单的激励机制。更普遍的做法是，很多近年来的研究将博弈论和机制设计中的概念应用于网络系统，通过最大化整个系统的性能来协调每个（对自身利益感兴趣的）用户的动机[18,37]。在本节，首先回顾一些基本的博弈论概念，然后对相关工作进行更全面的讨论。

7.2.1　博弈论背景

博弈论的基础之一是竞争均衡的概念，用于预测用户的行为和推断一个竞争博弈的结果。如参考文献［37］中所述，纳什均衡的概念主要用于对用户行为特性的系统研究。使用纳什均衡进行建模的一个吸引人之处，是它可以识别个体动机与其他参与者动机之间的张力[24]。

设每个用户具有一个效用（utility），其取值决定于该用户所采用的策略。当给定其他用户采取的一组策略时，纳什均衡定义为一组策略：如果用户希望最大化自己的效用，那么他们就必须选择这组策略[35]。

规范地说，对于以下简单形式的策略交互（称为博弈）：一个博弈中单个决策的制定者（也称为参与者）根据自己可用的策略选择其行为。在所有参与者行为的相互作用下，每个参与者将获得收益（payoff）。简单地说，每个参与者自行决定自己采取一个什么样的行为。基于自己和其他参与者的行为，该参与者得到奖励或惩罚。在经济学中，这种奖惩用一个一般化的术语“效用”来表示，有时也可以用一种货币的形式表示。在系统设计（如 CDN 设计）时，效用可以表示货币数量，当然也可以采用其他的形式表示。效用可以表示用户的满意度指标，该指标可以有各种定义方式，比如说它可以是一个下降的延时、增加的吞吐量或总体上更好服务的函数。

更准确地说，考虑一个参与者的集合 $V=\{1, \cdots, N\}$，用 Z_u 表示参与者 u 可用的所有纯策略（pure strategy，即确定的策略）的集合，Z_{-u} 表示除参与者 u 以外的其他参与者可用的策略集合，ζ_u 表示参与者 u 的策略集合中的任意一个策略。C_u 代表参与者 u 的收益（或效用）函数：给定参与者 u 的策略 ζ_u 和其他参与者的策略 ζ_{-u}，$C_u(\zeta_u, \zeta_{-u})$ 表示参

与者 u 的收益。一个有 N 个参与者的博弈可以描述为 $G=\{V;Z_u,Z_{-u};C_u,C_{-u}\}$。

当所有参与者采取一定的策略组合时，如果任意一个参与者的策略变化，都将导致其效用较变化前下降，那么所有的参与者处于一个纳什均衡[35]。一个纳什均衡定义为

定义 1：如果对于所有的 $u\in V$ 和 $\zeta_u\in Z_u$，$C_u(\zeta_u,\zeta_{-u}*)-C_u(\zeta_u*,\zeta_{-u}*)\leqslant 0$，则一个纯策略向量 $\boldsymbol{\zeta}*=(\zeta_1*,\cdots\zeta_N*)\in Z$ 构成博弈 G 的一个（纯）纳什均衡。

纳什均衡的概念可以扩展到概率化策略（probabilistic strategies）。确实存在这样的博弈：参与者采取的最优策略是非确定性的。例如，足球比赛中一个守门员在面对罚球时，总是向右扑救的策略很明显并非上策，因为对手很快就能发现这个规律。与其相反，大多数守门员会稍稍让他们的策略变得不确定，例如要是罚球手用左腿罚球他们就以 70% 的概率向右扑救。

注意，这种概率化的策略与完全随机的行为是不同的。在计算机系统中，概率化策略对于应对那些以概率形式来影响用户行为的外在条件是很有用的，如网络拥塞、无线信道中的噪声等。

具有一定概率分布的纯策略称为混合策略 σ_u。相应地，对于每一个参与者来说，其混合策略的集合 $\sum_u$ 包含了纯策略 Z_u 的集合（纯策略作为退化后的特例）。每一个参与者的随机选择都在统计意义上与其他参与者相互独立。一个由 N 个参与者组成的混合策略博弈可以被描述为 $G=\{V;\sum_u,\sum_{-u};C_u,C_{-u}\}$。

纳什均衡的概念可以引入到混合策略中。

定义 2：如果对于所有 $u\in V$ 和 $\sigma_u\in\sum_u$，都有

$$C_u(\sigma_u,\sigma_{-u}^*)-C_u(\sigma_u^*,\sigma_{-u}^*)\leqslant 0$$

则一个混合策略向量 $\boldsymbol{\sigma}^*=(\sigma_1^*,\cdots,\sigma_N^*)\in\sum$ 构成了博弈 G 的一个混合策略纳什均衡。

下面给出的第三个定义是所谓的“社会最优”。纳什策略在本质上揭示的是每个参与者面对其他所有参与者的策略时所能做出的最优反应，而社会最优描述的是对于所有参与者而言的最优，也就是将所有参与者作为一个整体来考虑。从本质上来说，纳什均衡是每个参与者针对其他参与者的策略做出的对自己最优的策略，而社会最优则就像是在一个“好心”的强权人物操纵下，使所有参与者采取了一种策略组合，在这种组合下所有参与者的平均收益是最高的。

回到上述公共牧场放羊的例子，每一个参与者（牧羊人）的纳什策略就是使他/她所有的羊使用牧场。然而，前面已经提到，所有的参与者都采纳这种策略的话，最终会导致灾难的发生。另外，一个社会最优策略能够强制性地限制每一个牧羊人能够在牧场上放牧的羊数，从而使所有的羊都能吃饱。

社会最优的定义如下：

定义 3：一个纯策略向量 $\boldsymbol{\zeta}^*=(\zeta_1^*,\cdots,\zeta_N^*)\in S$ 定义了博弈 G 的一个社会最优，当且仅当对于所有的 $\boldsymbol{\zeta}\in Z$，$\sum_u C_u(\zeta^*)-\sum_u C_u(\boldsymbol{\zeta})\leqslant 0$㊀

也就是说，社会最优就是能够最大化所有参与者效用的策略（或一组策略）。

㊀ 原文条件中的“≤”应该为“≥”。——译者注

7.2.2 博弈论在网络问题上的应用

近来，CDN 领域开始关注博弈论在算法和联网问题上的应用。参考文献［36］介绍了博弈论在算法问题上的应用状况。本节不再提供该方面的文献综述，而是主要讨论较少一部分的文献，它们代表了本研究领域的一些主流方向，能够为 CDN 设计者提供参考。具体而言，这里关注的是如下领域内的研究：网络路由、缓存技术和网络形成。

1. 流量的路由

通过研究一个特定的案例，Braess[6]的研究表明当添加一个新的路由来增加网络吞吐量时反而有可能会对整个网络的性能产生负面影响。在此之后，很多论文试图搞清路由数据的特性。例如，参考文献［43］通过设定边界来研究当用户能够自由选择如何路由他们的流量时网络状况会变得多差，借此泛化了 Braess 揭示的这种自相矛盾的现象。

其他文献对博弈论在路由方面的应用情况也进行了研究[8][19]，主要涉及如何在多跳网络中避免不希望发生的隐藏行为㊀。这个问题是指，在大多数多跳网络中，端节点只能确定端到端的传输是否成功，而一般很难监控到负责将报文进行路由的中间节点的行为。因此，如果不清楚中间层节点正常工作的内在原因，那么节点在路由流量的时候采取随意丢弃报文或者调低报文优先级的策略就会具有很大的风险。Feldman 等人[19]的研究表明，好的契约设计（contract design）对于解决隐藏行为是非常重要的。

2. 缓存和副本

很多论文（如参考文献［14，20，38］）研究了在分发网络中数据缓存和复制时的内在动机。在大尺度上，建立缓存的动机可以主要分为两类。对于一个 CDN 或覆盖网，需要确定数据存储在什么位置可以满足性能要求，同时又要避免一些缓存所在的节点产生逃离网络的动机。另外，由竞争对手所运营的不同的 CDN 可能需要互连，这会导致一个激励相容㊁的契约设计问题。

3. 网络形成

很大一部分研究工作[10,12,15,17,27]分析了网络形成的原因。也就是说，对于一个由存在着潜在竞争关系的理性实体组成的网络，这些文献研究了一个新的理性参与者是如何加入网络的。这个问题可以认为是一个典型的图问题，现有网络可以表达为一组节点的集合，这些节点通过一组边相连。由网络形成引出的问题可以被分解为如下一系列问题：在一个现有网络中新加入节点将与已有节点建立什么样的新连接（边）？当新节点接入网络后，会有某些边被取消吗？连接的创建可以被建模为一个动机问题：一个节点仅在能够增加其效用的情况下创建或者移除一个与其他节点的连接。

网络形成的代价模型起初被提出时是用于研究社会和经济网络的形成[27]，如确

㊀ 隐藏行为（hidden action）来自委托代理理论，该理论建立在信息不对称博弈的基础上，是指在某次活动中某些参与人拥有其他参与人不拥有的信息，故而背着其他人所做出的有利于自己利益的行为。——译者注

㊁ 激励相容（incentive - compatible）：来自 Hurwiez 建立的机制设计理论。每个理性人都会按照自利的原则行动，如果能设计一种制度，使每个人追求个人利益的行为，正好与集体价值最大化的目标相吻合，那么这种制度就是“激励相容”的。——译者注

定不同学者之间合作共写一篇论文的可能性。在这些社会、经济模型的基础上，近来的研究[14,17]分析了将对等网络形成作为非合作博弈的问题。在这个问题中，节点希望加入到网络中，同时也希望付出最小的代价。早期的研究工作关注的是节点之间的距离和连接度，将之作为计算每一个节点的效用函数的主要参数。之后的工作对早期的研究进行了拓展，除了上述距离和连接度外还考虑到了每一个节点上的负载[10,11]。在本章接下来的部分将详细介绍这些代价函数。

7.3 CDN 的代价和效益建模

本章的核心是一个代价模型，该模型的作用是评估当一个给定节点加入到 CDN 或覆盖网时，它必须付出的资源数量，以及它的加入给网络带来的效益。这里用一个开销下降的概念来表示加入 CDN 带来的效益，需要指出的是，该模型并不仅仅适用于 CDN，事实上它可以应用于存在以下行为的任何网络中：节点请求资源和提供资源服务，或者对其他节点之间的请求进行服务。这样的网络除了 CDN 之外，也包括 P2P 文件共享系统[2,16]、ad - hoc 网络[39]、分布式检索服务（distributed lookup services）[42,48]、应用层组播覆盖网（application - layer multicast overlays）[5,13,30]等。表 7.1 中列出了该模型中使用的主要符号。

严格地说，可以使用一个四元组（V,E,K,F）来定义一个网络（CDN、覆盖网等）。其中，V 是网络中的节点集，E 是有向边集，K 是网络中相关资源的集合，F: $K \to V$ 是将资源分配到节点的映射函数。

每一个节点 $u \in V$ 被分配一个唯一的标识（整数或者字符串），为简单起见用 u 来表示。定义 $K_u = \{k \in K: F(k) = u\}$ 为存储在节点 $u \in V$ 中的资源集合，有 $K = \cup_u K_u$，且不失一般性假设集合 K_u之间不相交。事实上，如果一个资源被存储在多个节点处（复制），那么这些副本可被视为不同的资源，且被请求的概率完全相同。

表 7.1 符号说明

符号	说明
内容属性	
K_u	节点 u 拥有的资源集合
F	内容到节点的映射函数
X	发出请求的源节点
Y	被请求的资源
网络属性	
V	网络中节点的集合
E	网络中边的集合
K	网络中资源的集合
N	节点个数
D	维度
Δ	基
网络测量	
$t_{u,v}$	节点 u 和 v 之间的跳数
$\chi_{v,w}(u)$	测试 u 是否在节点 v 至 w 的路径上

符号	说明
开销测量和函数	
L_u	节点 u 的延时开销
R_u	节点 u 的路由开销
S_u	节点 u 的服务开销
M_u	节点 u 的维护开销
$l_{u,k}$	节点 u 请求资源 k 的标称开销
$r_{u,k}$	节点 u 转发资源 k 的标称开销
$s_{u,k}$	节点 u 提供资源 k 的标称开销
$m_{u,v}$	节点 u 维护节点 v 的信息的标称开销
C_u	节点 u 的总开销
C	网络的总开销

通过两个相互独立的随机变量 $X \in V$ 和 $Y \in K$ 来分析每一个请求，它们分别表示发出请求的节点 X 和被请求的资源 Y。

对于一个给定节点 $u \in V$，每当一个资源 $k \in K$ 在整个网络中被请求时，节点 u 将处于如下四种状态之一。

1. 空闲

节点 u 不拥有 k 或者没有请求 k，也并不处于请求路由路径上。节点 u 没有任何代价问题。

2. 分发请求

节点 u 请求资源 k。在本章的模型中，用减少等待时间的形式来表示加入一个 P2P 网络的收益，这与参考文献［17］等提出的方法类似。特别地，假定持有资源 k 的节点 v 离 u 距离越远（从拓扑意义上而言），请求所付出的开销越高。如果在节点 u 和 v 之间没有通路，那么请求无法执行，这会产生无穷大的开销。更确切地说，这里将 u 请求 k 产生的开销建模成为 $l_{u,k}t_{u,v}$，其中 $t_{u,v}$ 是从 u 到 v 之间的跳数，而 $l_{u,k}$ 是（正）比例因子。定义节点 u 的延时开销 L_u 为个体开销 $l_{u,k}t_{u,v}$ 乘以 k 被请求的概率，即

$$L_u = \sum_{v \in V} \sum_{k \in K_v} l_{u,k} t_{u,v} \Pr[Y = k] \tag{7.1}$$

其中，如果没有从节点 u 到 v 的路径，那么 $t_{u,v} = \infty$，且对于任意 u，有 $t_{u,u} = 0$。使用该定义，为了避免无穷大的开销，每一个节点都有创建连接的动机，以便与其他持有其感兴趣资源的节点之间存在路径。另一种方案是在本地存储或缓存所有感兴趣的资源，这样所有请求的开销可降至 $l_{u,k}t_{u,u} = 0$。

下面给出一个延时开销的具体例子：域名解析服务（DNS）[33]。DNS 可以被视为一个使用树形拓扑的覆盖网，其中叶节点是 DNS 用户，其他节点是 DNS 服务器。假设一个用户 u 需要访问 DNS 上的一个记录 k，如果 k 之前被访问的极少，以至于请求需要被转发到 DNS 根服务器 v，这时可能会得到一个从 u 到 v 之间的很高的跳数，如 $t_{u,v} = 5$。在请求被解析之后，u 的主 DNS 服务器 u' 将具有 k 的一个副本，这就将从 u 到 v 的请求延时开销从 $t_{u,v} = 5$ 降低到 $t_{u,u'} = 1$。引入权重 $l_{u,k}$ 表示不同资源 k 之间的相对重要程度，式（7.1）中 u 的延时表示为 u 所有请求延时的加权平均。在这个 DNS 的例子中，从节点 u 的角度来说，如果解析 k = www. google. com 的能力比解析 k' = dogmatix. sims. berkeley. edu的能力高 100 倍，那么就应该使 $l_{u,k} = 100 l_{u,k'}$。

3. 服务请求

节点 u 拥有资源 k，且对请求进行服务的开销为 $s_{u,k}$。例如，在文件共享系统中，节点提供文件服务需要占用一定的上传带宽。定义服务开销 S_u 为 $s_{u,k}$ 在所有可能请求下的数学期望，即

$$S_u = \sum_{k \in K_u} s_{u,k} \Pr[Y = k]$$

回到前面的 DNS 例子，在 DNS 服务器 u' 中添加一条关于 k 的记录意味着 u' 需要付出一些存储资源，这里的模型将其表达为服务开销 $S_{u'}$ 的一个增量。在 DNS 的例子中，对于一个给定的 DNS 服务器，对 DNS 服务器中记录 k 的服务开销与所有其他的记录都是一样的，因此对于所有 k，$s_{u',k} = s_{u'}$，这对应于存储一个记录的代价。

4. 转发请求

节点 u 并不持有或者请求 k，但是需要将对 k 的请求转发出去，因此需要付出开销 $r_{u,k}$。对于经过节点 u 的所有可能的请求路径，它需要付出的总的路由开销 R_u 为转发所有资源 k 的开销 $r_{u,k}$ 的平均值。也就是说，对于 $(u,v,w)\in V^3$ 有二值函数

$$\chi_{v,w}(u)=\begin{cases}1, & \text{如果 } u \text{ 位于从 } v \text{ 到 } w \text{ 的路径上(不含 } v \text{ 和 } w\text{)}\\ 0, & \text{其他}\end{cases}$$

R_u 的表达式为

$$R_u = \sum_{v\in V}\sum_{w\in V}\sum_{k\in K_*} r_{u,k}\Pr[X=v]\Pr[Y=k]\chi_{v,w}(u) \tag{7.2}$$

以 DNS 为例，路由开销表示：对于所有可能的请求，一个服务器收到对 k 的请求但无法解析 k 的位置，从而将请求转发到 DNS 树的更高级服务器时，该服务器所使用的资源的平均值。

另外，每一个节点都维护某些状态信息使得网络控制协议能够正确运转。在大多数覆盖网协议中，每一个节点 u 都需要维护一个邻接表并与所有邻节点交换信息，也就是说，如果节点 v 是 u 的邻节点，则存在一个彼此相连的边 (u,v)。用 $\mathcal{N}(u)$ 表示 u 的邻节点集合，定义维护开销 M_u 为

$$M_u = \sum_{v\in\mathcal{N}(u)} m_{u,v}$$

式中，$m_{u,v}\geqslant 0$ 表示节点 u 为了维持与邻节点 $v\in\mathcal{N}(u)$ 的连接所产生的开销。

回到 DNS 的例子，当 DNS 服务器 u 收到的请求不能在本地服务时，u 为了能与其附近的其他服务器进行通信而需要维护相关的信息，维护开销即为 u 维护相关信息所使用的资源。

最后，定义节点 u 付出的总代价函数 C_u 为

$$C_u = L_u + S_u + R_u + M_u$$

可以使用 C_u 来计算整个网络的开销，即 $C=\sum_{u\in V}C_u$。需要注意的是，只有 S_u，R_u，M_u 和 L_u 都使用相同的单位时，C_u 的表达式才有意义，也就是说 $s_{u,k}$，$r_{u,k}$，$l_{u,k}$ 和 $m_{u,v}$ 等系数需要正确设定。例如，如果 $l_{u,k}$ 按照每个资源每跳使用货币单位给出，那么 $m_{u,v}$ 也需要使用货币单位来表示。

7.4 社会最优和纳什均衡

本节主要探讨能够通过上面定义的代价模型达到一个社会最优或纳什均衡状态的网络结构。规范地说，将一个网络结构定义为一个“几何”[22]，即与路由选择机制相关的由一些节点和边构成的集合（拓扑结构）。除非另有说明，均默认使用最短路径路由，并区分不同的拓扑结构。在讨论一些可能的社会最优之前，将使用一些简化手段，这有助于使分析得到简化。在分析过程中，主要关注结论和本质，大多数技术证明的细节问题不占用过多篇幅，建议感兴趣的读者可参见参考文献［11］获得一个更全面的了解。

假设：在本节的剩余部分，以一个含有 $N(N>0)$ 个节点的网络作为分析对象，

其中对于所有的 $u \in V$ 和 $k \in K$，$l_{u,k}=l$，$s_{u,k}=s$，$r_{u,k}=r$，同时对于所有的 $u \in V$ 和 $v \in V$，$m_{u,v}=m$。也就是说，假定所有节点在一个单跳延时、响应一个请求、路由一个请求或者维护一个连接上的开销完全相同，与资源被请求还是被服务无关。一般来说，该假设较为粗糙，但是对于一个由同质（homogeneous）节点构成、包含固定大小的资源以及同质连接的网络而言这样的简化还是足够精确的（尤其是对于基于分布式哈希表的检索机制来说，例子可参见参考文献［40，42，44，48，49］）。

假设网络处于稳态，也就是说，节点不会加入或者脱离网络，此时 l、s、r 以及 m 的值都是常数。同时，假设请求在所有节点的集合上服从均匀分布，即对于任何节点 u，有 $\Pr[X=u]=1/N$。

在分析中，做了进一步的简化：选择映射函数 F 使得所有的节点都以相同的概率对请求进行服务，即 $\sum_{k \in K_u}\Pr[Y=k]=1/N$，这意味着

$$S_u=\frac{s}{N}$$

也就是说，与网络的几何结构无关。在7.6节进行的数值仿真中将取消该假设。另外，如果使用 $E[X]$ 来表示变量 X 的数学期望，则式（7.1）和式（7.2）分别退化为

$$L_u=lE[t_{u,v}]$$

$$R_u=rE[\chi_{v,w}(u)]$$

另外，因为每一个节点 u 的邻节点数为 $\deg(u)$，所以可以得到

$$M_u=m\deg(u)$$

最后，假设所有的节点都不会出现恶意行为。

7.4.1 全网状网

对于可能的社会最优，首先对全网状网（Full Mesh）进行分析。在一个全网状网中，每一对节点都通过双向边连接，即对于任意节点 $v \neq u$，$t_{u,v}=1$。全网状网中节点之间的流量和路由都是相同的，因为任意两个节点之间总有一条直连的边，并且 $\deg(u)=N-1$。这样，对于所有 u，有 $R_u=0, L_u=l(N-1)/N, M_u=m(N-1)$。由 $S_u=s/N$，有 $C_u=s/N+l(N-1)/N+m(N-1)$。对 u 求和，可得

$$C(\text{全网状网})=s+l(N-1)+mN(N-1) \tag{7.3}$$

现在开始从全网状网中移除一条连接，如节点 0→1 的连接。此时节点0的维护开销 M_0 降低了 m。不过，为了访问节点1上的资源，节点0就需要通过其他节点发送请求。节点0经由哪个节点将请求发送到节点1的实际机制与本讨论无关。例如，不失一般性可以假定节点周期性地从自己的邻节点列表中选择。假定节点0选择节点2，那么 L_0 增加了 l/N，且节点2的路由开销 R_2 增加了 r/N^2。

因此，移除节点0至节点1的连接后总开销的变化为 $\Delta C=-m+l/N+r/N^2$，如果 $\Delta C \geqslant 0$，那么移除连接意味着总开销的增加，此时全网状网是社会最优。特别地，有如下命题：

命题1：全网状网是社会最优的条件为维护代价“足够小”，即

$$m \leqslant \frac{l}{N}+\frac{r}{N^2} \tag{7.4}$$

注意，当 $N \to \infty$ 时式（7.4）中的 m 趋于0。事实上，使用一阶泰勒级数展开式，可以将 $\Delta C \geqslant 0$ 作为影响 N 的条件：当 $m << l^2/r$ 时，$N \leqslant \lfloor l/m + r/l \rfloor$。

在此也可以结合DNS对式（7.4）进行说明。由式（7.4），如果互联网主机的数量维持在合理的较少数量上，那么每一个主机就可以借助一个很大的HOSTS. TXT文件来直接将主机名解析为IP地址，从而有效地为命名覆盖创建一个全网状网：每一个节点都知道所有其他的节点[⊖]。只有在互联网主机的数量增加到在单个、分布式文件上维护所有信息变得不切实际时，才引入DNS协议。

7.4.2 星形网络

现在假设式（7.4）并不成立，并且网络结构为一个星形拓扑。令 $u = 0$ 表示星形网络（Star Network）的中心，该中心节点对所有外围节点的流量进行路由。也就是说，对于所有的节点 $v \neq w(v,w>0)$，$\chi_{v,w}(0) = 1$。显然，$R_0 = r(N-1)(N-2)/N^2$，$L_0 = l(N-1)/N, M_0 = m(N-1)$。这时，星形网络的中心所产生的开销 C_0 为

$$C_0 = m(N-1) + \frac{s}{N} + l\frac{N-1}{N} + r\frac{(N-1)(N-2)}{N^2} \tag{7.5}$$

外围节点并不路由任何流量，即对于所有的 $u>0$，有 $R_u = 0$。并且，所有的外围节点与星形网络中心节点的距离为1，与其他 $N-2$ 个节点的距离为2，这时有 $L_u = l(2n-3)/N$。另外，对于所有的外围节点有 $\deg(u) = 1$。这时，$M_u = m$，同时节点$u>0$的单个开销为

$$C_u = m + \frac{s}{N} + l\frac{2N-3}{N} \tag{7.6}$$

命题2：只有当 N 是常数或 $l = r = m = 0$ 时，$C_0 = C_u$成立。

证明：假设 $C_0 - C_u = 0$。因为 $N \neq 0$，所以 $C_0 - C_u = 0$ 等价于 N^2（$C_0 - C_u$）$=0$。使用式（7.5）和式（7.6）中 C_0和 C_u的表达式，将条件 N^2（$C_0 - C_i$）$=0$ 改写为 N 的多项式形式，并且乘以因子 $N-2$，可以得到

$$(N-2)(mN^2 - (l-r)N - r) = 0$$

N 的多项式恒等于0，当且仅当多项式的所有系数均为0，因此 $l = r = m = 0$。

对于星形网络，中心节点和其他节点的开销存在着较大的差异，$C_0 - C_u$可以从理论上对激励（阻碍）一个节点成为中心节点的因素进行量化。所以，从命题2可知，对大多数情况而言在网络中心存在着一个激励（阻碍）节点成为中心节点的因素。在实际中，星形布局的网络代表了高度中心化的结构，资源都集中在一个“汇聚点”。这个汇聚点一般位于内容提供商处，它作为一个独一无二的服务器并凭借其中心地位，可以得到适当的补偿（如资金上的投入）。

下面计算星形网络的整体开销，并确定在什么条件下该网络是社会最优。对式（7.5）和式（7.6）求和，可得

⊖ 注意：这里只关注域名解析，HOSTS. TXT文件的更新和传播是另一个单独的问题，实际上是以一种集中的方式完成的[33]。——原书注

$$C(\text{星形网络}) = 2m(N-1) + s + 2l\frac{(N-1)^2}{N} + r\frac{(N-1)(N-2)}{N^2} \tag{7.7}$$

命题3：对于任意的节点数 $N \geq 3$，星形网络是社会最优的条件为：式（7.4）不成立，且所有连接是双向的，即对于任意节点 $u \in V$，$v \in V$，如果 $(u \to v) \in E$ 则 $(v \to u) \in E$。

命题3的证明可参见参考文献［11］。该证明的核心是反复从一个全网状网中移除连接，直到网络近乎断开。由此表明，星形配置实际上最大化了所有参与者的整体效用。

关于命题3有两点值得注意。

首先，命题3并没有保证星形网络是一个唯一的社会最优。事实上，当 $m = l/N + r/N^2$ 时，在星形网络的外围节点之间加入任意数量的"捷径"也能实现社会最优。

其次，对于任意的 N，双向连接这一假设对于命题3的成立至关重要。例如，如果允许存在单向连接，那么可以推出，如果 m 足够大且 N 很小，则单向环 $0 \to 1 \to \cdots \to N \to 1$ 的开销要低于星形网络。

然而，在允许单向连接的情况下寻求社会最优是一个尚未解决问题，星形网络在实际应用中仍然会占据主要地位，而单向环这种拓扑结构可能仅仅使用于限制性很强的条件下。更简单地说，以上分析表明，当需要维护的连接数量太多以至于全网状网不再是具有吸引力的解决方案时，从资源消耗的角度可以认为一个中心化的网络是最优的。

7.4.3 纳什均衡

现在假定每一个节点都可以选择自己需要保持的连接，但是对于自身具有的资源不采取任何控制，同时响应所有路由请求。换句话说，每一个节点在建立连接时都是自私的，但是一旦连接建立起来节点就不再这样。当每一个节点 u 具有（完全）理性时，即在已知其他节点行为的前提下试图最小化自身的开销 C_u，其结果是形成的拓扑结构成为了一个纳什均衡，如7.2节的定义。

虽然在一般情况下不能保证纳什均衡的存在性或唯一性，但存在以下均衡：

命题4：如果 $m < l/N$，全网状网是一个唯一的（纯）纳什均衡。

命题5：如果 $m \geq l/N$，星形网络是一个纯纳什均衡。

以上结论可以通过下面的事实验证：根据命题4的条件，移除全网状网的连接会降低至少一个用户的效用值。反之，根据命题5的条件，向星形网络中增加的一条连接会使至少一个节点的效用值降低[11]。

命题4和命题5说明，如果维护连接的成本不高或网络结构很小，那么唯一的纳什均衡就是全网状网；如果维护连接的成本很高或网络结构很大，那么星形网络是一个可能的纳什均衡，且不能保证纳什均衡的唯一性。例如，当 $m = l/N$ 时，通过在星形网络的外围节点之间加入任意数量的连接所构成的任何网络都会构成一个纳什均衡。

上述结论是参考文献［27］中结论的一个泛化，该文献使用的是一个不同的代价模型，没有使用路由开销。

7.4.4 解释

上述结论在图 7.1 中进行了汇总，图中根据单位维护开销 m 的值来区分社会最优和纳什均衡。

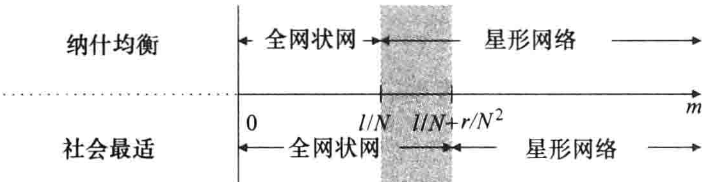


图 7.1 社会最优和纳什均衡（在区间 $[l/N, l/N+r/N^2]$，单个节点的动机与社会最优失调）

当 $m<l/N$ 时，全网状网既是纳什均衡也是社会最优；当 $m>l/N+r/N^2$ 时，星形网络既是纳什均衡也是社会最优。两种情况下，每一个节点的动机都与网络整体资源最高效的利用目标相一致，见图中的白色区域。

图 7.1 中最值得注意的是灰色区域，在这个区域中个体的动机与高效利用整体资源的目标是相互冲突的。该区域对应于参数区间 $l/N<m<l/N+r/N^2$，其大小只取决于 r。换句话说，如果假设所有节点都以相同的概率对请求进行服务，那么一旦节点很在乎它们帮助其他节点转发流量所占用的自身资源，那么社会最优可能会与纳什均衡明显背离。

从上述分析可知，如果存在一个“转发免费”的网络（也就是 $r=0$，如带宽和计算能力无限廉价），那么这个网络从系统设计者来看就是理想的，因为在这种情况下所有的个体动机就应该可以形成一个社会最优的解决方案。然而不幸的是，在大多数网络中转发数据的成本是不容忽略的。这表明，相对于以前模型仅考虑基于节点度（如维护开销）和跳数（如延时开销），这里的代价模型更加适合于捕获节点可能的消极动机。

对于 CDN 设计者而言，这个结果产生了一个很重要的结论，即要么 CDN 被设计为尽可能避免让节点转发过多的数据；要么，如果前者不可能的话，转发节点就应该能根据它们路由的流量，按照一定的计算方法得到相应的补偿。

7.5 现有结构的分析

在前面的讨论中，忽略了对于网络攻击的鲁棒性、容错能力或潜在的性能瓶颈。所有这些因素都会对中心化的方法带来实际的挑战，使节点产生占据（或逃离）星形结构中心位置的动机。使用全网状网能够在很大方面上避免这些问题，但是就如我们已经看到的那样，全网状网仅仅是一个针对网络节点数量适中情况下的解决方案。

目前的很多研究工作都集中在设计网络的几何布局，以实现合理的性能和解决上面提到的鲁棒性问题。在本节中，使用前面提出的代价模型对一些路由几何结构进行

评估，这些几何结构是在近来的建网文献中针对覆盖网架构提出的。本节主要分析的是 de Bruijn 图、D 维 Torus 网（D – dimensional Torus）、PRR 树和 Chord 环，给出了一个节点在以上不同几何结构中的各种代价值。与前面一样，本节关注的是结果和结果背后的原因，更多的技术细节读者可以参见参考文献［11］。随后，将结果与前面对社会最优和纳什均衡的研究结果进行比较。

7.5.1 de Bruijn 图

凭借一些吸引人的性质，如短平均路由距离及对节点失效的高恢复性，de Bruijn 图常用于各种覆盖网的维护算法[28,31,34]。在一个 de Bruijn 图中，任意节点 u 用标识符（u_1，…，u_D）表示，其中符号 u_i（$i=1$，…，D）来自一个大小为 Δ 的符号表。对于符号表中的任意一个符号 x，从节点（u_1，…，u_D）出发有一条有向边连接到其他节点（u_2，…，u_D，x）。在由此产生的有向图中，所有的节点都有固定的出度 Δ，且有向图的直径为 D。

设 V' 为一个节点集，其中每一个节点的标识符具有（h，h，…，h）的形式。由于 V' 中的每个节点都有一条从自己出发并回到自己的有向边，所以对于所有的 $u\in V'$，维护开销 $M_u=m(\Delta-1)$。对于不在 V' 中的节点 u，$M_u=m\Delta$。下面两个引理表明，每一个节点的路由开销也依赖于节点在图中的位置。

引理 1：在最短路径路由下，节点 $u\in V'$ 并不路由任何流量，且 $R_u=0$。

这个引理可以通过参考文献［11］的反例得证。

引理 2：经过一个给定节点 u 的路由数（节点负载）ρ_u 存在上界 $\rho_u\leqslant\rho_{\max}$，其中

$$\rho_{\max}=\frac{(D-1)[\Delta^{D+2}-(\Delta-1)^2]-D\Delta^{D+1}+\Delta^2}{(\Delta-1)^2}$$

该边界是严格的，对于节点（0，1，2，…，$D-1$），当 $\Delta\geqslant D$ 时可以达到该边界。

引理 2 的证明在本质上与参考文献［47］中的证明方法相似，即限定通过给定边的最大路由数量，完整的证明可参见参考文献［11］，在此简单介绍如下。

首先，确定经由节点 u 的长度为 k 的路径数的上界，这等价于计算长度为 $D+k$ 的串（u 的标识符作为其子串）的最大数量。接着，在 $k\in$［1，D］区间上对 k 求和，并得到一个中间界；通过移除长度为 $2D$ 的所有串来进行优化。长度为 $2D$ 的串代表了一个环，而环不能确定 de Bruijn 图中的最短路径。

通过引理 1 和引理 2 可以推出，在一个 de Bruijn 图中，对于任意的节点 u、v、w，有 $0\leqslant\Pr[\chi_{v,w}(u)=1]\leqslant\rho_{\max}/N^2$。因为 $\chi_{v,w}(u)$ 是一个二值函数，所以 $\Pr[\chi_{v,w}(u)=1]=E[\chi_{v,w}]$。最终得到 $0\leqslant R_i\leqslant R_{\max}$，其中

$$R_{\max}=\frac{r\rho_{\max}}{N^2}$$

下面给出延时开销的上界和下界，即 $L_{\max}$ 和 $L_{\min}$。根据参考文献［11］，节点 $u\in V'$ 满足条件 $L_u=L_{\max}$，其中

$$L_{\max}=l\,\frac{D\Delta^{D+1}-(D+1)\Delta^D+1}{N(\Delta-1)}$$

下界为

$$L_{\min}=\frac{l}{N}\left(D\Delta^{D}+\frac{D}{\Delta-1}-\frac{\Delta(\Delta^{D}-1)}{(\Delta-1)^{2}}\right)$$

且当$\Delta\geqslant D$时，对于节点$(0,1,\cdots,D-1)$，有$L_u=L_{\min}$。

注意，当$N=\Delta^D$，即所有的标识符都已分配给节点时，$L_{\min}$和$L_{\max}$的表达式可以进一步简化。

7.5.2 D 维 Torus 网

在D维 Torus 网中，每一个节点由D个笛卡尔坐标表示，且有$2D$个邻节点。对于任意节点u，维护开销为$M_u=2mD$。此类型的路由几何结构的一个应用是 CAN[42]。

每一个节点上路由的实现方法，都是基于“贪婪”的方式，即根据与目标节点欧氏距离最短的原则转发到其邻节点。这里假设每个节点在D维空间中均匀分布。这个约束也可以表达为稍强一点的约束：假设$N^{1/D}$是整数，且笛卡尔坐标集$(u_1, \cdots, u_D)$（其中每个u_i都映射为区间$[0, N^{1/D}-1]$上的一个整数）的任何一种取值都映射到某个节点，即假设标识符空间$(u_1, \cdots, u_D)$完全分配。

由参考文献［31］可知，一个路由路径的平均长度当N为偶数时为$(D/4)N^{1/D}$跳，当N为奇数时为$(D/4)N^{1/D}+D/4-o(1)$。因为假设D维 Torus 网为均匀分区，由对称性可得：对于所有的u，使用与参考文献［42］相同的近似，即路由路径的平均长度在N为奇数时近似等于$(D/4)N^{1/D}$跳，有

$$L_u=l\frac{DN^{1/D}}{4}$$

为了确定路由开销R_u，计算节点负载时将其作为维数D的函数$\rho_{u,D}$。根据上述D维 Torus 网平均分区的假设，由对称性可知$\rho_{u,D}$对于所有的u都是相等的。

引理 3：在一个充满节点且节点数为N的D维 Torus 网中，任意节点u的负载为

$$\rho_{u,D}=1+N^{\frac{D-1}{D}}\left\{-N^{\frac{1}{D}}+D\left[N^{\frac{1}{D}}-1+\left(\left\lfloor\frac{N^{\frac{1}{D}}}{2}\right\rfloor-1\right)\left(\left\lceil\frac{N^{\frac{1}{D}}}{2}\right\rceil-1\right)\right]\right\} \tag{7.8}$$

该引理的证明可以对维数D使用归纳法[11]。首先，对于$D=1$，在资源固定分布的情况下，每个节点u的负载$\rho_{u,1}$等于经过所有节点的路由数之和。例如当$N=7$时，对于任意节点u，有$\rho_{u,1}=0+1+2+2+1+0=6$。由此可得N为奇数和偶数时的两个不同的表达式，其一般化的形式为

$$\rho_{u,1}=\left(\left\lfloor\frac{N}{2}\right\rfloor-1\right)\left(\left\lceil\frac{N}{2}\right\rceil-1\right)$$

当$D>1$时，计算经过每个节点u的路由数$\rho_{u,D}$的关键在于，因为一条路径上的两个相邻节点的坐标差异不会超过一个维度，所以沿着笛卡尔坐标系存在着若干条等效的最短路径。例如对于$D=2$，从节点$(0, 0)$到节点$(1, 1)$：$\mathscr{P}_1=(0,0)\to(1,0)\to(1,1)$和$\mathscr{P}_2=(0,0)\to(0,1)\to(1,1)$是等效的最短路径。因此，总是能够选择一条以正确的顺序改变坐标的路径——从第一维坐标开始变化，也就是上例中的$\mathscr{P}_1$。

设 v 为路由的起始节点，w 为目标节点，节点 u，v，w 的坐标分别用（u_1，…，u_D）、（v_1，…，v_D）和（w_1，…，w_D）表示。根据路由算法，中间节点 u 每次改正一个坐标时仅有 3 种可能：①节点 u 与源节点 v 和目标节点 w 具有相同的第 D 维坐标，即 $u_D = v_D = w_D$；②节点 u、v 和 w 的第 D 维坐标均不相同，即 $u_D \neq v_D \neq w_D$；③节点 u 和节点 v 具有同样的第 D 维坐标，但与目标节点 w 不同，即 $u_D = v_D$，$u_D \neq w_D$。通过计算每一种情况的节点负载并求和，可得式（7.8）中 $\rho_{u,D}$ 的值。

对于所有的节点 u，通过 $\rho_{u,D}$ 可以立刻得到

$$R_u = r\frac{\rho_{u,D}}{N^2}$$

7.5.3　PRR 树

下面分析 PRR 树的变种[40]，其应用可参见 Pastry[44] 或 Tapestry[49]。节点表示为以 Δ 为基的 D 个数字的串（u_1，…，u_D）。对于 $i = 1 \cdots D$ 且 $x \neq u_i \in \{0, \cdots, \Delta - 1\}$，每一个节点都以（$u_1$，…，$u_{i-1}$，$x$，$y_{i+1}$，…，$y_D$）的形式连接到 $D(\Delta - 1)$ 个不同的邻节点，此时维护开销为 $M_u = mD(\Delta - 1)$。

对于余下坐标 y_{i+1}，…，y_D 的各种可能取值，协议一般会通过一个邻近度（proximity metric）来选择一个附近的节点。假设节点在空间中的分布是均匀的，且所有标识符已全部分配，则可以选择 $y_{i+1} = u_{i+1}$，…，$y_D = u_D$。这样，相隔距离为 n 跳的两个节点 u 和 v 在坐标中相差 n 个数。

显然，有 $\binom{D}{n}$ 种不同的方式来选择差异出现在哪些数，而且每个差异的数都有 $\Delta - 1$ 个可能的取值。因此，对于一个给定节点 u，与之相隔距离为 n 的节点有 $\binom{D}{n}(\Delta - 1)^n$ 个，再乘以节点的总数 $N = \Delta^D$，并除以路径总数 N^2，可以推出：对于所有 u、v、w，有

$$\Pr[t_{u,v} = n] = \frac{\binom{D}{n}(\Delta - 1)^n}{N} \tag{7.9}$$

对于任意两个满足 $t_{u,v} = n$ 的节点 u、v，因为路由是唯一的，所以在 u 和 v 之间的路径上存在 $n - 1$ 个不同的节点。因此，随机抽取一个节点 w，它位于 u 至 v 的路径上的概率为

$$\Pr[\chi_{u,v}(w) = 1 \mid t_{u,v} = n] = \frac{n - 1}{N} \tag{7.10}$$

对式（7.9）和式（7.10）使用全概率公式，将等号右边表示为展开式的导数的函数，结合 $\Pr[\chi_{u,v}(w) = 1]$ 并使用二项式定理，然后乘以 r，可得路由开销

$$R_u = r\frac{\Delta^{D-1}[D(\Delta - 1) - \Delta] + 1}{N^2} \tag{7.11}$$

使用式（7.9）中给出的 Pr $[t_{u,v}=n]$ 并利用二项式定理[11]，访问开销 L_u 为

$$L_u = lE[t_{u,v}] = l\sum_{n=1}^{D} k\Pr[t_{u,v} = n] = l\frac{D\Delta^{D-1}(\Delta - 1)}{N} \tag{7.12}$$

注意，对于 $N=\Delta^D$，式（7.12）退化为 $L_u=lD(\Delta-1)/\Delta$。

7.5.4 Chord 环

在一个 Chord 环[48]中，节点用二值串表示（即 $\Delta=2$）。当环的全部标识符完全分配时，每个节点 u 与 D 个邻节点相连，这些邻节点的标识符为 $[(u+2^p) \bmod 2^D]$，其中 $p=0$，…，$D-1$。与对 PRR 树的分析类似［式（7.11）和式（7.12）］，可得 R_u和 L_u（$\Delta=2$ 时）。仿真结果验证了本节的分析[48]。

7.5.5 讨论

本节的分析结果可以用于很多情况。首先，这些结果表明，本节中提及的各种路由几何结构都具有同样的渐近特性：路由开销以 $\log N$ 的数量级下降，延时开销以 $\log N$ 的数量级增长。其次，虽然这些渐进性众所周知（参见参考文献［22，31，42，48］），但本节博弈论分析的主要优点在于能提供闭式的方程，可以在实际应用中根据对不同开销（如路由开销和延时开销）的相对偏重，调整配置参数 Δ 或 D。第三，本节的分析提供了一个基线，用于与以下方面进行比较：①社会最优和（或）纳什均衡；②一些更实际的网络状况，如网络中标识符空间稀疏分配或者节点中的某些资源需求相对更大。这些对比分析构成了下一节的内容。

7.6 定量评估

本节使用蒙特卡洛仿真对上述不同网络结构的分析结果进行比较。另外，上面的分析中使用了一些假设，如所有资源都具有相同的访问几率、标识符全部分配。本节将放宽这些约定，对放宽后的影响进行定量的分析，使上节的分析更加完善。

1. 与社会最优进行比较

首先给出 7.5 节的数值分析结果。如表 7.2 所示，给出了 5 个具有不同 Δ 和 D 取值的 de Bruijn 图，且请求源 X 和被请求资源 Y 为相互独立的服从均匀分布的随机变量。从表中的数据可以看出，当所有节点的延时开销相差不大时，R_{max}和次优情况下的路由开销㊀R'_{min}的比值一般较大。因此，如果 $r>>l$，则那些 $R_u=R_{max}$的节点可能会存在“换座”的动机。例如，这些节点可能会离开网络，然后再很快回来，希望能被分配一个不同的标识符 $u'\neq u$，以减轻自己的负载。这就可能需要增加一些机制，如增加一个进入网络的成本，来避免节点的这种离开行为。

㊀ 即除集合 V'中 Δ 个节点外的所有其他节点的 R_u 的最小值（对于 V'中的节点，$R_u=0$）。——原书注

表 7.2　de Bruijn 图中开销的不对称性（$l=1$，$r=1\ 000$）

(Δ, D)	L_{min}	L_{max}	$\frac{L_{max}}{L_{min}}$	R'_{min}	R_{max}	$\frac{R_{max}}{R'_{min}}$
(2, 9)	7.18	8.00	1.11	3.89	17.53	4.51
(3.6)	5.26	5.50	1.04	2.05	9.05	4.41
(4, 4)	3.56	3.67	1.03	5.11	13.87	2.71
(5, 4)	3.69	3.75	1.02	1.98	5.50	2.78
(6, 3)	2.76	2.80	1.01	5.38	9.99	1.86

现在对上述不同网络几何结构下的开销进行了仿真。选择 $\Delta=2$，此条件对 PRR 树和 Chord 环的效果是完全相同的。对 D 维 Torus 网，选择 $D=\{2, 6\}$，对其他几何结构选择 $D=\log_{\Delta}N$。

在仿真中，节点数量在 $N=10$ 到 $N=1\ 000$ 之间变化。对每一个 N 的取值，以不同的种子值进行 10 次蒙特卡洛仿真，每次包含 100 000 个请求，且参数使用不同的值，X 和 Y 为相互独立服从均匀分布的随机变量。对所有的请求和节点求平均，得到的延时开销和路由开销如图 7.2 所示。

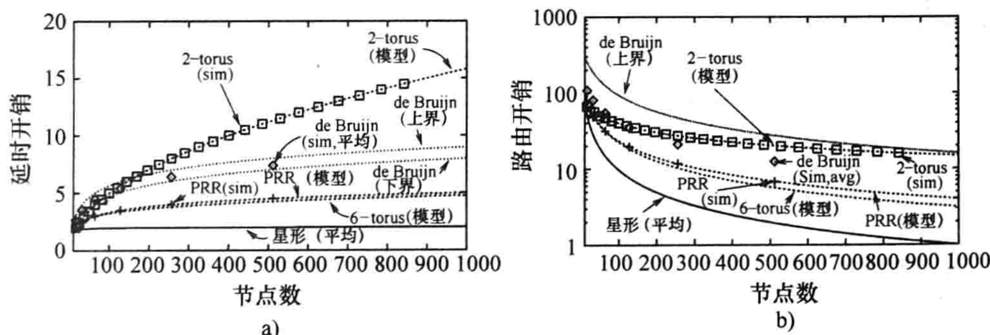


图 7.2　延时开销和路由开销

a）延时开销（$l=1$）　b）路由开销（$r=1, 000$）

（注：标有 sim 的曲线表示仿真结果，由于全网状网的延时开销恒为 1，路由开销恒为 0，故这两项结果因显示问题在图中略去）

由图 7.2 可知，星形网络的平均开销较其他几何结构低，这与之前的分析相吻合，即在很多情况下星形网络都是社会最优的。但需要注意的是，本节的代价模型并未考虑扩展性或者恢复性因素，而这两个因素都是完全中心化的结构中需重点考虑的因素。另外，虽然星形网络是潜在的纳什均衡，但仍需要一定的激励机制（如货币奖励）来对星形网络中存在的节点开销不对称问题进行补偿，并鼓励一个节点在第一步就去占据中心位置。

2. 资源访问的不对称性

下面将验证把"所有资源都具有相同的访问几率"这一假设放宽后，对之前分析结果的影响。为此进行了一组仿真，受参考文献［7］的启发，仿真中假设用户对资源的访问模式服从 Zipf－like 分布。

在这组蒙特卡洛仿真中，网络的节点数设定为 $N=512$。对于 D 维 Torus 网，选择 $D=3$；对于其他几何结构的网络，选择 $\Delta=2$ 和 $D=9$。仿真运行了 1024 次，每次包含 100 000 个请求。请求源 X 是服从均匀分布的随机变量，被请求的资源 Y 服从

Zipf - like分布，有

$$S_u = s\Omega / [\text{Rank}(u)]^\alpha$$

式中，$\alpha = 0.75$，$\Omega = (\sum_{i=1}^{N} i^\alpha)^{-1}$，Rank：$V \rightarrow \{1, \cdots, N\}$ 是一个双射函数，根据节点 u 持有给定资源的概率，对节点 $u \in V$ 按照概率递减的原则给出 u 的排序值。

因为 Y 不再是一个服从均匀分布的随机变量，因此不同的节点具有不同的延时开销和路由开销。在每一个试验中，对所有节点计算了延时开销和路由开销的最大值（L_{max} 和 R_{max}）和最小值（L_{min} 和 R'_{min}）之间的比率。在 de Bruijn 图中，因为有些节点并不路由任何流量，所以再次令 $R'_{min} = \min_{u \in V} \{R_u > 0\}$。

在表 7.3 中显示了 1024 次试验的平均比率 L_{max}/L_{min} 和 R_{max}/R'_{min}。括号中的数表示标准差。结果表明，对于任何几何结构，所有节点上的延时开销比较相似，但路由开销则有显著的差异。在 de Bruijn 图中更高的不对称性可以通过节点负载的差异来解释（参见 7.5 节），因为节点负载的差异放大了路由开销的差异。与社会最优相比，在星形网络或者全网状网中，路由开销和延时开销相似，而与不同资源的受欢迎程度无关。

表 7.3 当资源请求概率服从 Zipf - like 分布时，网络中开销的不对称性

	$\frac{L_{max}}{L_{min}}$	$\frac{R_{max}}{R'_{min}}$
3 - Torus	1.2675 （±0.0442）	5.2845 （±0.3516）
de Bruijn	1.2453 （±0.0265）	30.7275 （±9.5970）
PRR 树	1.2591 （±0.0420）	9.2154 （±0.6590）

下面确定路由开销中的不对称性是否对延时开销（即服务开销）中的不对称性实现了补偿。为此，本节分别计算了 R 与 L、R 与 S 及 L 与 S 之间的相关系数（对于两个变量 x 和 y，记为 Corr（x，y）)，计算数据包括可用于三元组（R，L，S）的 512 个节点 ×1 024 次试验 = 524 288 个数据点，计算结果见表 7.4。从表 7.4 可以看出，对于这三种几何结构，S 与 R 之间及 S 与 L 之间几乎不存在相关性[⊖]。换句话说，服务开销对 R 或 L 几乎没有影响。R 和 L 之间的相关度也非常低，表明不同的节点可能最终会具有完全不同的开销。

对于这 3 种路由几何结构，资源请求的非对称性可能会导致不同节点存在完全不同的开销。开销的差异导致一些节点会过载，或者至少会使节点产生很强的动机先离开然后再重新加入网络，以便能得到一个"更好的位置"。这个结果强调了，如果 CDN 使用的协议取决于某个上述的路由几何结构，那么就必须意识到协议中具有高效的负载均衡原型的重要性。

表 7.4 资源访问服从 Zipf - like 分布的网络中路由、延时和服务开销之间的相关性

	Corr（R，L）	Corr（R，S）	Corr（L，S）
3 - Torus	-0.3133	-0.0166	-0.0960
de Bruijn	-0.3299	-0.0112	-0.0981
PRR 树	-0.2278	-0.0128	-0.1027

⊖ 相关系数实际上只用于测试线性相关。一般需要使用其他的测试（如 η 测试或相关比）来确定两个变量之间不存在相关性。这里不再进行此类测试，需要说明的是，附加数据（如散布图）可以确定变量之间不存在相关。——原书注

3. 标识符空间稀疏分配

至此，本文一直假设标识符空间是完全分配的。例如，在一个 PRR 树中，如果$\Delta=2$、$D=9$ 则必然包含 $N=512$ 个节点。而事实上，标识符空间有可能是稀疏分配的。

在下面的仿真实验中，对于每一种几何结构，本文使用固定数量的节点 $N=512$。首先，从完全分配的标识符空间开始进行仿真。对于 de Bruijn 图和 PRR 树，设 $\Delta=2$、$D=9$，并逐步增加 D 直到 $D=15$。对于 D 维 Torus 网，令 $D=3$，使每个节点 u 都由一组坐标（u_x，u_y，u_z）表示，其中每个坐标 u_x、u_y 和 u_z 的取值都在 0 到 n 之间。开始，选择 $n=8$ 使任何一组坐标都对应于一个给定节点（因为 $n^D=N$）。接着，逐步增加 n 直到 $n=32$，即对于所有的三种拓扑结构，标识符空间中的标识符数从 512 增加到 32768 个。最初没有映射到任何节点的那些标识符用一个服从均匀分布的随机变量来进行选择。对于 D 的每一个取值（分别对应于 n 的所有取值），使用不同的随机种子进行 100 次 Monte Carlo 仿真实验，对应于标识符的 100 种不同的分配方式。每一个实验包括 100000 个请求，实验中 X 和 Y 是相互独立服从均匀分布的随机变量。

对于每一种几何结构，图 7.3 中给出了由 100 次实验（对应于一个给定的标识符个数）结果计算出的比率 R_{max}/R'_{min}、L_{max}/L_{min} 和 M_{max}/M_{min} 的均值，以及这三个比率在最坏情况（即最大值）下的值。

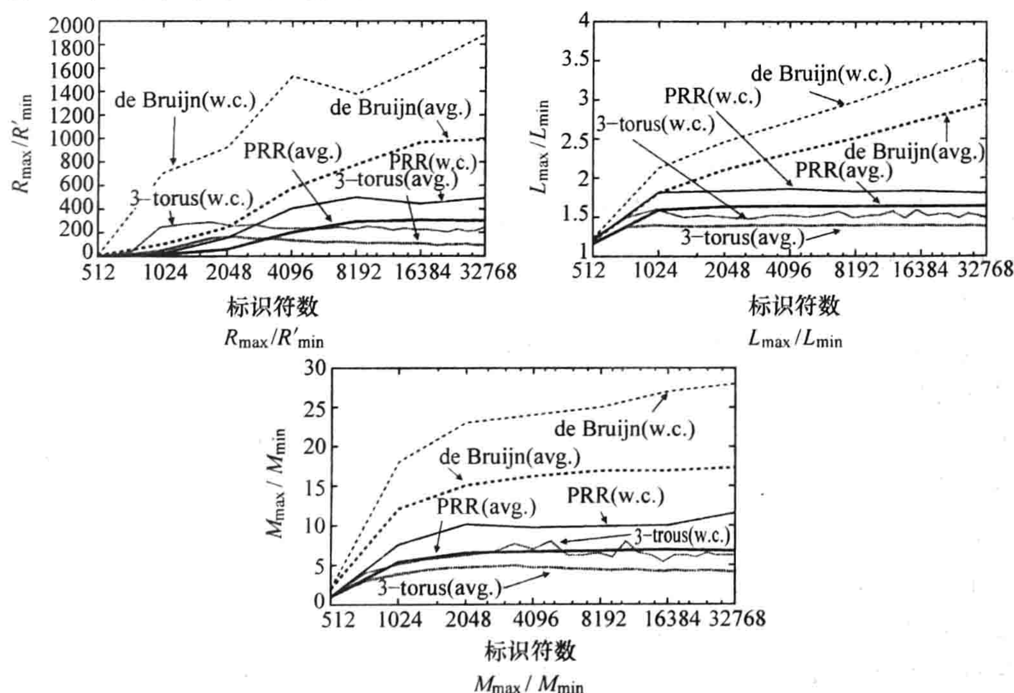


图 7.3　使用不同个数的标识符时，一个给定节点的路由、延时及维护开销的最大值与最小值之间的比率

（注：标有“avg.”的曲线表示给定集上所有实验的平均结果，而标记“w. c.”表示最坏情况，即最大比率）

实验结果显示，在一个稀疏分配的标识符空间中，延时开销的不平衡性保持相对温和（因子值在 3 ~ 4 左右）；而维护开销的不平衡性表现得较为明显（从最高到最低时相应的比率在 20 ~ 25 左右）。最主要的发现是，路由开销的不平衡性可能会变得

非常大，从最好到最坏情况时比率在几百到几千之间变化。这个结果强调了对高效负载均衡路由算法的迫切需求，如果没有这样的路由算法，那么大量的节点可能会有这样的强烈冲动：离开网络并随即重新加入网络，以换到一个更好的位置，这就导致了不稳定性的存在。

此外，实验结果还表明，标识符空间的稀疏分配会使各种不同的开销之间变得相关[11]。这个结果印证了之前的猜测，即一个给定节点的路由和延时开销很大程度上取决于节点与网络中其他节点之间连接的好坏程度，而这恰恰又是通过维护开销来体现的。

7.7 写给使用者

到目前为止，在对 CDN 中相关各方的理性行为进行建模时，纳什均衡被认为是一个非常适合的工具，其主要优点在于它的简单性。不过，纳什均衡对每个博弈参与者的能力和目标施加非常严格的假设，使其能够预测那些与直觉或现实相悖的结果。特别是，一般认为理性参与者应该能做出正确的决策、对博弈具有完美的预测以及无限的计算能力。从直觉上说，网络参与者可能会偏离上述严格的假设：对于网络用户来说，他们不可能具备非常高的理性；又如自动代理，它们则可能会由于软件缺陷、配置不正确或计算资源的不充分等原因而出现故障。

因此，有充分的理由来给出更一般化的策略行为模型，而将纳什均衡的概念作为其特例。特别是，为了放宽纳什均衡所要求的“完全理性”这一假设，有人提出了有限理性（bounded rationality）的概念。有限理性的参与者在整个决策空间中不需要选择最好的策略，而是允许在很多方面上出现误差，如评估与一个策略相关的收益、评价最优可用策略或对某个策略的执行等方面。

目前存在很多不同的技术对有限理性进行建模。一个方法是将（可能是少量的）噪声引入到制定决策的过程中（如参考文献［21］）。另一种有限理性的均衡模型是随机最优反应均衡（Quantal Response Equilibrium[32]，QRE）。如果用户计算给定策略的相关收益时存在着误差，则可以使用 QRE 对博弈中均衡的特性进行分析。

也许最简单的松弛方法就是近似理性（near rationality）[4,41]，如 ε - 均衡的概念[41]。ε - 均衡是将一个完全理性参与者的概念放宽得到这样的一个模型：在该模型中每一个参与者满足于接近（不必取得）对其他参与者策略的最优响应，一个参与者不论采取什么其他策略，其效用的提高也不会超过 ε（$\varepsilon>0$）。这样，通过确定每个参与者的策略来找到一个 ε - 均衡，使得在给定其他参与者的策略时，一个参与者的收益与其最大可能收益的差不超过 ε。

定义 4：如果对于所有的 $i\in N$、$\sigma_i\in\Sigma_i$ 及一个固定大小的 $\varepsilon>0$，有 $u_i(\sigma_i,\boldsymbol{\sigma}^{\varepsilon}_{-i})-u_i(\sigma^{\varepsilon}_i,\sigma^{\varepsilon}_{-i})\leqslant\varepsilon$，则混合策略向量 $\boldsymbol{\sigma}^{\varepsilon}=(\sigma^{\varepsilon}_1,\cdots,\sigma^{\varepsilon}_N)\in\boldsymbol{\Sigma}$ 构成一个博弈 G 的混合策略 ε - 均衡。

如果一个纯策略的向量 $\boldsymbol{\zeta}^{\varepsilon}\in Z$ 满足上述均衡条件，那么它就构成一个纯策略 ε - 均衡。当 $\varepsilon=0$ 时，就退化为纳什均衡这个特例。这样，可以将 ε - 均衡视为竞争均衡的一个更一般化的解。

1. 覆盖网

如之前的命题 4 和命题 5 所示，如果只考虑纳什均衡，则当参数 m、l 和 N 设定为一定的取值组合时，星形网络或全网状网最有可能实现纳什均衡。由纳什均衡扩展到 ε - 均衡，如果有 $m \in [l/N - \varepsilon, l/N + \varepsilon]$，则无论网络拓扑结构为何均可以构成一个 ε - 均衡。该结论的证明并不复杂，只需简单地将 ε 加入到命题 4 和命题 5 的证明中即可。另外，有损信道等原因会导致网络连接的失效。如果节点为解决该问题而使用混合策略而不是纯策略，那么若想使得任何网络都是一个 ε - 均衡，估计 m 的可能取值范围要比 2ε 大得多。

2. 实际应用问题

在均衡概念本身存在着不确定性的情况下，将博弈论应用到 CDN 设计可以得到多少好处呢？

首先，姑且不论均衡的概念，社会最优分析对于理解网络性能的上限是至关重要的。Papadimitriou 定义了一个有意思的性能测度，称之为无政府的代价（price of anarchy）[37]。无政府的代价是纳什均衡在最坏情况下的总效用 $\sum_u C_u$ 与社会最优的比率，作为系统设计时一个有用的度量，它可以显示出在参与者各自为政的情况下性能到底可以降低到什么程度。

其次，尽管存在严格的假设，纳什均衡仍不失为一个非常有用的工具，可以作为个体动机的一个一阶近似。特别地，在纳什均衡中参与者的行为会导致最坏情况的发生，这就有助于设计者通过分析获得网络性能的下界。TCP 拥塞控制的例子可以证明纳什均衡可能是一个最坏情况下的均衡：在一个 TCP/IP 网络中，如果那些竞争带宽的参与者达到了一个纳什均衡，那么它们都会关闭拥塞控制（或使用 UDP）[3]。但实际情况远远不是这样，因为大多数用户还是乐于不修改其 TCP 参数，这可以用有限理性的观点来解释。

再有，使用近似理性（而不是完全理性）可以有助于评估一个博弈模型的精度。如果模型缺乏鲁棒性，那么它成为现实情况下一个精确模型的机会就会降低。在上述例子中可以看到，输出不确定结果的参数空间随着不确定量 ε 而线性增长，这表明模型足够鲁棒，也很可能是可靠的。另外，其他的一些非组网的例子[12]并没有显示出这种性质。

7.8　未来研究方向

本章讨论的是应用博弈论对覆盖网（如 CDN）中的交互进行建模。不过，用经济学原理指导网络设计并不限于博弈论，更一般化的方法是对网络中个体动机的研究。因此，这个研究领域将来应该会继续保持活力。

在短期内，本章的研究成果估计会有助于开发出更精致成熟的动机型。值得注意的是，近来系统设计方面越来越多的文献[14][26]利用了经济学原理来构建其方案。

更明显的趋势是，考虑到内容分发市场的快速发展、设施提供商之间激烈的竞争以及潜在的微薄利润，CDN 的设计者为了确保收益将不得不更好地理解市场的力量和个体的动机。

另一个潜在的研究方向是通过对动机的研究来加强 CDN 的安全性。例如，参考

文献［29］显示，通过重构现有的技术来实现对存在着分歧的个体动机进行重组，可以建立一个安全性更高的覆盖网，帮助那些处于潜在竞争的内容提供商提高面对DDoS攻击时的网络恢复能力。

这个研究预示了一个令人感兴趣的趋势：之前面临的问题已经完全不同了。未来的系统不再试图解决动机协调的问题，而是可能会深入研究各种动机之间的差异，从而通过让不同的实体在网络中充当最适合它们自身目标的角色，来实现更优的系统设计。

7.9 结论

基于负载和节点连通性，本章提出了一个代价模型，用来解决一个覆盖网（如CDN）中的节点开销问题。这个代价模型对于不同网络拓扑性能的测量是一个有用的补充，因为它一揽子地实现了以下目标：预测妨碍协作的因素（即节点为了降低开销而拒绝对请求进行服务）、发现可能的网络不稳定性（即节点为了降低开销而离开或重新加入网络）、识别热点（具有高路由负载的节点）及描述一个网络的效率。

前面说过，一个节点为其他节点转发流量时会占用其自身的资源。本章的一个关键结论是，如果节点很在乎这些被占用的资源，就可能会产生低效问题。在这种情况下，让节点根据自己的意愿来选择保持哪些网络连接，那么对于整个资源的使用来说就能得到一个次优网络。

分析还显示，必须设计非常有效的负载均衡原语来避免节点之间开销的不平衡，以消除这种不平衡给网络带来的潜在不稳定性。除了负载均衡原语机制，一个可能的选择是动机机制，它使节点乐于转发尽可能多的网络流量。本章给出的有关博弈论的公式有助于激励相容的设计。

最后，本章提出的框架可以帮助一个CDN的设计者确定哪一类拓扑结构更适于所处的特定环境，如在ad－hoc网上的内容分发、视频点播等。这个努力将成为一种尝试，即参数化地评估各种拓扑之间相对的标称开销。

致谢

本章出现的一些资料来自IPTPS'04、INFOCOM'05和PINS'04[10,11,12]，文中的研究工作主要是在Nicolas Christin就职于加州大学信息学院期间进行的。本章的研究受国家科学基金（ANI－0085879和ANI－0331659）的部分资助。本章部分内容的形成得益于与Paul Laskowski的讨论。

参考文献

[1] Napster protocol specification (2000). http://opennap.sourceforge.net/napster.txt
[2] Adar, E., Huberman, B.: Free riding on Gnutella. First Monday **5**(10) (2000)
[3] Akella, A., Seshan, S., Karp, R., Shenker, S., Papadimitriou, C.: Selfish behavior and stability of the Internet: A game-theoretic analysis of TCP. In: Proc. ACM SIGCOMM'02, pp. 117–130. Pittsburgh, PA (2002)

[4] Akerlof, G., Yellen, J.: Can small deviations from rationality make significant differences to economic equilibria? American Economic Review **75**(4), 708–720 (1985)
[5] Banerjee, S., Bhattacharjee, B., Kommareddy, C.: Scalable application layer multicast. In: Proc. ACM SIGCOMM'02, pp. 205–217. Pittsburgh, PA (2002)
[6] Braess, D.: Über ein Paradoxon aus der Verkehrsplanung. Unternehmensforschung **12**, 258–268 (1969)
[7] Breslau, L., Cao, P., Fan, L., Philips, G., Shenker, S.: Web caching and Zipf-like distributions: Evidence and implications. In: Proc. IEEE INFOCOM'99, pp. 126–134. New York, NY (1999)
[8] Buchegger, S., Le Boudec, J.Y.: Performance analysis of the confidant protocol. In: Proc. ACM MobiHoc'02, pp. 226–236. ACM (2002)
[9] Cha, M., Kwak, H., Rodriguez, P., Ahn, Y.Y., Moon, S.: I Tube, You Tube, Everybody Tubes: Analyzing the world's largest user generated content video system. In: Proc. ACM IMC'07. San Diego, CA (2007)
[10] Christin, N., Chuang, J.: On the cost of participating in a peer-to-peer network. In: Proc. IPTPS'04, Lecture Notes in Computer Science, Vol. 3279, pp. 22–32. San Diego, CA (2004)
[11] Christin, N., Chuang, J.: A cost-based analysis of overlay routing geometries. In: Proc. INFOCOM'05, Vol. 4, pp. 2566–2577. Miami, FL (2005)
[12] Christin, N., Grossklags, J., Chuang, J.: Near rationality and competitive equilibria in networked systems. In: Proc. ACM SIGCOMM'04 Workshop on Practice and Theory of Incentives in Networked Systems (PINS), pp. 213–219. Portland, OR (2004)
[13] Chu, Y.H., Rao, S., Zhang, H.: A case for endsystem multicast. In: Proc. ACM SIGMETRICS'00, pp. 1–12. Santa Clara, CA (2000)
[14] Chun, B.G., Chaudhuri, K., Wee, H., Barreno, M., Papadimitriou, C., Kubiatowicz, J.: Selfish caching in distributed systems: a game-theoretic analysis. In: Proc. ACM PODC'04, pp. 21–30. Saint John's, NL, CA (2004)
[15] Chun, B.G., Fonseca, R., Stoica, I., Kubiatowicz, J.: Characterizing selfishly constructed overlay networks. In: Proc. IEEE INFOCOM'04, Vol. 2, pp. 1329–1339. Hong Kong (2004)
[16] Cohen, B.: Incentives build robustness in BitTorrent. In: Proc. 1st Workshop on the Economics of Peer-to-Peer Systems. Berkeley, CA (2003)
[17] Fabrikant, A., Luthra, A., Maneva, E., Papadimitriou, C., Shenker, S.: On a network creation game. In: Proc. ACM PODC'03, pp. 347–351. Boston, MA (2003)
[18] Feigenbaum, J., Shenker, S.: Distributed algorithmic mechanism design: Recent results and future directions. In: Proc. DIAL-M'02, pp. 1–13. Atlanta, GA (2002)
[19] Feldman, M., Chuang, J., Stoica, I., Shenker, S.: Hidden-action in multi-hop routing. In: Proc. ACM EC'05, pp. 117–126. ACM (2005)
[20] Goemans, M., Li, L., Mirrokni, V., Thottan, M.: Market sharing games applied to content distribution in ad-hoc networks. In: Proc. ACM MobiHoc '04, pp. 55–66. ACM, Roppongi Hills, Tokyo, Japan (2004)
[21] Goeree, J., Holt, C.: A model of noisy introspection. Games and Economic Behavior **46**(2), 365–382 (2004)
[22] Gummadi, K., Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S., Stoica, I.: The impact of DHT routing geometry on resilience and proximity. In: Proc. ACM SIGCOMM'03, pp. 381–394. Karlsruhe, Germany (2003)
[23] Hardin, G.: The tragedy of the commons. Science **162**(3859), 1243–1248 (1968)
[24] Holt, C., Roth, A.: The Nash equilibrium: a perspective. Proc. National Academy of Sciences **101**(12), 3999–4002 (2004)
[25] Hsieh, H.Y., Sivakumar, R.: Performance comparison of cellular and multi-hop wireless networks: A quantitative study. In: Proc. ACM SIGMETRICS'01, pp. 113–122. Cambridge, MA (2001)
[26] Huang, C., Li, J., Ross, K.: Can internet video-on-demand be profitable? In: Proc. ACM SIGCOMM'07, pp. 133–144. Kyoto, Japan (2007)
[27] Jackson, M., Wolinsky, A.: A strategic model for social and economic networks. Journal of Economic Theory **71**(1), 44–74 (1996)

[28] Kaashoek, M.F., Karger, D.: Koorde: A simple degree-optimal distributed hash table. In: Proc. IPTPS'03, pp. 323–336. Berkeley, CA (2003)
[29] Khor, S.H., Christin, N., Wong, T., Nakao, A.: Power to the people: Securing the Internet one edge at a time. In: Proc. ACM SIGCOMM'07 Workshop on Large-Scale Attack Defense (LSAD), pp. 89–96. Kyoto, Japan (2007)
[30] Liebeherr, J., Nahas, M., Si, W.: Application-layer multicast with Delaunay triangulations. IEEE Journal of Selected Areas in Communications **20**(8), 1472–1488 (2002)
[31] Loguinov, D., Kumar, A., Rai, V., Ganesh, S.: Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience. In: Proc. ACM SIGCOMM'03, pp. 395–406. Karlsruhe, Germany (2003)
[32] McKelvey, R., Palfrey, T.: Quantal response equilibria for normal form games. Games and Economic Behavior **10**(1), 6–38 (1995)
[33] Mockapetris, P., Dunlap, K.: Development of the domain name system. In: Proc. ACM SIGCOMM'88, pp. 123–133. Stanford, California (1988)
[34] Naor, M., Wieder, U.: Novel architectures for P2P applications: the continuous-discrete approach. In: Proc. ACM SPAA'03, pp. 50–59. San Diego, CA (2003)
[35] Nash, J.: Non-cooperative games. Annals of Mathematics **54**(2), 286–295 (1951)
[36] Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (eds.): Algorithmic Game Theory. Cambridge University Press (2007)
[37] Papadimitriou, C.: Algorithms, games and the Internet. In: Proc. ACM STOC'01, pp. 749–753. Heraklion, Crete, Greece (2001)
[38] Pathan, A.M., Buyya, R.: Economy-based content replication for peering content delivery networks. In: Proc. CCGRID, pp. 887–892. IEEE Computer Society (2007)
[39] Perkins, C. (ed.): Ad hoc networking. Addison-Wesley, Boston, MA (2000)
[40] Plaxton, C.G., Rajamaran, R., Richa, A.: Accessing nearby copies of replicated objects in a distributed environment. Theory of Computing Systems **32**(3), 241–280 (1999)
[41] Radner, R.: Collusive behavior in noncooperative epsilon-equilibria of oligopolies with long but finite lives. Journal of Economic Theory **22**, 136–154 (1980)
[42] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proc. ACM SIGCOMM'01, pp. 161–172. San Diego, CA (2001)
[43] Roughgarden, T., Tardos, É.: How bad is selfish routing? Journal of the ACM **49**(2), 236–259 (2002)
[44] Rowston, A., Druschel, P.: Pastry: Scalable, decentralized object location and routing for large scale peer-to-peer systems. In: Proc. IFIP/ACM Middleware'01, pp. 329–350. Heidelberg, Germany (2001)
[45] Saroiu, S., Gummadi, K., Gribble, S.: A measurement study of peer-to-peer file sharing systems. In: Proc. SPIE/ACM MMCN'02, pp. 156–170. San Jose, CA (2002)
[46] Shenker, S.: Making greed work in networks: A game-theoretic analysis of switch service disciplines. IEEE/ACM Transactions on Networking **3**(6), 819–831 (1995)
[47] Sivarajan, K., Ramaswami, R.: Lightwave networks based on de Bruijn graphs. IEEE/ACM Transactions on Networking **2**(1), 70–79 (1994)
[48] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup protocol for Internet applications. IEEE/ACM Transactions on Networking **11**(1), 17–32 (2003)
[49] Zhao, B., Huang, L., Stribling, J., Rhea, S., Joseph, A., Kubiatowicz, J.: Tapestry: A resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications **22**(1), 41–53 (2004)

第8章 CDN的定价

Kartik Hosanagar

8.1 引言

CDN是内容分发供应链的一个重要组成部分。这个供应链包括创建内容的内容提供商、骨干网、用来传输内容的接入网络以及那些存储信息并将信息从网络边缘分发到网络终端用户的CDN提供商。通过在网络边缘附近实现服务器的同地协作（co-locating），CDN在这个利益链上具有独特的地位，因此被互联网中大量的主要内容提供商使用。随着越来越多的人使用宽带，加上内容分发不断向多媒体方向发展，CDN将在互联网的媒体分发中扮演越来越重要的角色。

CDN给内容提供商带来了明显的经济利益。通过在网络边缘处分发内容，CDN使网络内容的分发更加迅捷。不仅如此，CDN可以通过整合多个用户站点的流量，在网络设施成本方面实现规模经济效益。因此，CDN内容分发的边际成本将远远小于那些只在自己专有网络上进行内容发布的小内容提供商。流量的整合也降低了用户对内容的需求差异所带来的压力。一般来说，不同内容提供商传输内容时的流量不会在同一时刻达到峰值，所以困扰单个内容提供商的突发性流量峰值问题可以通过增加其CDN设施来解决。最后，由于CDN可以使用多台备选服务器提供服务，所以单个节点一般不会成为瓶颈，这就有助于改善内容的可用性，特别是在瞬时拥塞和遭到DoS攻击的时候。

与此同时，CDN在设施的维护和内容分发上也会带来成本问题。考虑到CDN引入的成本和内容提供商可能获得的巨大收益，需要一个市场机制来保证不仅内容提供商能够继续获得收益，CDN提供商也要有部署和管理其设施的积极性。最简单的机制就是给CDN服务定价，这就是本章的关注焦点。

CDN定价问题已在业内得到了相当大的关注。定价策略明显影响到CDN提供商和内容提供商的利润，各方关注的问题包括突发流量对定价和收益的影响[7]、当前定价方案中使用的总量折扣㊀和超额问题[18]以及竞争和价格战对CDN市场的影响[15]等。

随后的各节安排如下：首先对业界常用的定价模型进行回顾；随后，介绍包括CDN定价在内的一些对内容分发市场经济效益的学术研究；在8.3节中给出了一个模型，

㊀ 总量折扣（volume discount）是指，如果消费者的购买数量达到某个约定值，则全部商品（不同于超额折扣中的超额部分）按约定的折扣计价。例如，衬衫单价100元，而一次购买3件的价格为240元。——译者注

用于分析内容提供商从 CDN 服务中获得的收益，并使用该模型研究定价策略；接下的章节将对行业中的参与各方进行分析；最后，提出未来的研究方向和本章的结论。

8.2 业界常用的定价模型

CDN 使用了很多不同的定价模型，这些模型都有其独到之处。下面简要介绍两个最流行的定价模型。

8.2.1 基于总量的定价

最简单的定价方法是针对流量达到某个特定级别的内容提供商（如每月 50TB）。基于约定的不同流量级别，CDN 按照每 GB 设定价格。这些定价方案通常提供一个总量折扣（例如，流量在 40 ~ 50TB 时价格为 0.5 元/GB，100TB 以上时为 0.15 元/GB）。不过，当内容提供商的流量超过每月定额时，一些 CDN 提供商会实行一个惩罚措施，其目的是引导内容提供商更精确地估计每月流量数据，这反过来也有助于 CDN 进行更好的设施容量规划。不过，总量折扣一般都是针对那些流量非常高的内容提供商。

8.2.2 基于百分位数的定价

另一个常用的定价方式是基于流量的第 95 百分位数[⊖]。在这个策略中，CDN 对内容提供商的带宽使用情况进行周期性的采样，在月末计算带宽使用率是否超过 95%，并在此基础上根据带宽使用情况以 Mbit/s 为单位收取费用。

大多数 CDN 提供商都提供上述两种方案，内容提供商可以选择对自己合适的方案。在本章的后面部分将讨论这些不同定价方案的差异。

8.3 背景和相关工作

目前，有两大类工作与 CDN 的定价密切相关。第一类是对存在着拥塞的网络进行定价研究；第二类是从经济性角度研究内容分发，其中包括一些对 CDN 定价的研究。

8.3.1 网络中的拥塞定价

对网络拥塞的定价研究通常着眼于定价与服务质量（QoS）之间的相互作用，采用的方法是研究拥塞成本和网络容量成本之间的折中。当网络容量不能简单地增加且网络应用对 QoS 的需求非常严格时，定价就对实现期望的 QoS 起到了关键的作用。特别地，价格的提高可以促使用户降低他们的流量并控制对网络服务的要求，这就反过

⊖ 例如，将一个班级的考试成绩从低到高排列，如果一个学生的成绩比 95% 的同学高，则其成绩就是第 95 百分位数。——译者注

来降低了网络的拥塞。Mendelson[16]、Mendelson 和 Whang[17]针对存在延时开销的计算机服务提出了相应的定价模型，其他的相关工作包括对互联网中存在拥塞的资源进行定价[5,14]以及在基于 QoS 方案按优先级传输数据包时的 QoS 定价，如 Diffserv 和 Int-serv[2,6]。

有的参考文献研究了在流量突发情况下的定价问题。Kelly 认为，流量突发时的定价可以基于有效带宽[13]。不过，有效带宽依赖于网络连接资源和流量多路复用的特性，因此并不容易预测。为此，Kelly 提出了一个近似解，包括按单位时间、单位流量和单个连接等几种方式的收费[13]。

在 CDN 的语境中，定价的目标并不是单纯为了降低 CDN 内的拥塞，内容提供商不会因为流量达到预设的限制而去拒绝客户端发出的请求。例如，瞬时拥塞不能事先预测，并且内容提供商购买 CDN 服务就是为了应对流量的峰值问题，而这恰恰就是 CDN 的魅力所在。由于其用户之间的流量不是高度相关的，所以一个内容提供商流量的激增可以通过分配给其更多一些资源而轻松解决。因此，拥塞的降低并不是定价的关键目标。这样，即使拥塞定价的研究成果能用得上，也并非就顺理成章地应用到 CDN 领域。

8.3.2 内容分发的经济性

CDN 的历史可以追溯到代理服务器使用的缓存，当时 ISP 零售商用之实现在网络边缘对内容进行存储和分发。Hosanagar 等人[8,9]从经济学角度研究了网络缓存技术，发现当内容提供商转向动态内容发布并同时搜集关于网站使用习惯的精确信息时，传统的“尽力而为”（best - effort）缓存技术的采用率将会降低。Hosanagar 等人认为，CDN 服务的重要作用体现在充当下面两方的媒介：寻求边缘分发收益的内容提供商和有能力在网络边缘处安装服务器的 ISP 零售商。从这个角度看，CDN 允许内容提供商在避免“尽力而为”缓存技术带来的成本的前提下，获得边缘分发的收益。

Kaya 等人[12]对内容分发市场进行了一个经济上和操作上的分析。针对 CDN 的定价问题，Ercetin 等人[4]研究了一个区分服务（differentiated service）CDN 的最优定价和资源分配方案，Hosanagar 等人[7,10]研究了一个垄断的 CDN 提供商的最优定价方案。他们发现，当内容提供商具有相似的流量突发水平时，传统的基于使用量的定价方案需要使用总量折扣的策略。不过，当内容提供商流量突发的原因并不一样时，这种折扣被证明是次优的。进一步，他们发现基于百分位数的定价策略的收益可能大大高于传统的基于使用量的收费方法。在下节中，将更详细地讨论上述研究中的模型及其效果。

除此之外，还存在着其他的内容边缘分发模型，包括基于 P2P 的内容分发。Courcobetis 等人[3]研究了基于 P2P 的内容分发的市场模型，将其作为基于 CDN 的内容分发的替代物。Johar 等人[11]研究了 P2P 网络对 CDN 市场的影响。他们发现，由于在 P2P 网络中内容提供商的资源被非法地分发，P2P 网络能够促使内容提供商寻求高质量的 CDN 服务，将之作为与上述非法行为竞争的一个手段，这反而可能会使 CDN 企业有可能从中受益。

8.4 CDN 价格模型

本节提出的 CDN 定价模型，可以分析内容提供商从 CDN 服务中获得的利益，这些模型基于参考文献［10］，用于研究定价策略。

假设一个市场中有一个垄断的 CDN 提供商，这时内容提供商面临两个选择：要么自行分发内容（通过自己的服务器或者托管服务，以下简称为自给方式），要么外包给一个 CDN。内容提供商之间的差异在于它们的外包成本 C_o 和平均流量水平（即平均抵达率）λ。平均抵达率是内容提供商处理流量大小的一个度量。以往的经验性研究[9]显示，平均抵达率等于 λ 的内容提供商的数量为 $g(\lambda)=\beta/\lambda^{\delta}$，其中 $\delta\in[1,2]$ 且 β 为常数。将内容分发进行外包的成本 C_o 包括为便于 CDN 分发而对内容进行修改的成本，以及与第三方（即 CDN 提供商）共享秘密数据的成本。假设 C_o 的累积分布函数（cdf）和概率分布函数（pdf）分别由式 $H(C_o)=C_o^W$ 和式 $h(C_o)=WC_o^{W-1}$ 给出，其中 W 是一个正的常数，$C_o\in[0,1]$。调整参数 W 可以调节外包成本低的那些内容提供商的相对比例。当 $W=1$ 时，C_o 服从均匀分布；当 $W>1$ 时分布向负方向倾斜，当 $W<1$ 时分布向正方向倾斜。实际中，C_o 的上界可以是任意值，但不失一般性本节将其规格化为 1。一般而言，因为 CDN 可以直接观测到全部的流量信息，所以 CDN 知道所有内容提供商的 λ，但是 CDN 并不清楚内容提供商的外包成本 C_o。这里，假设所有内容提供商外包成本的分布 $H(C_o)$ 已知。

为了确定 CDN 的最优定价机制，本节按照如下方式进行分析：首先，确定一个内容提供商在自给和通过 CDN 分发内容两种情况下的期望收益。在确定这两种情况下的预期收益后，内容提供商从中选择收益较高的那种。购买 CDN 服务的决策是一个定价策略的函数。根据内容提供商的购买决策函数，可以确定使 CDN 预期收益最大的最优定价策略。下面分析内容提供商在自给和借助 CDN 分发内容这两种情况下的期望收益。

8.4.1 内容提供商采用自给方式

对于一个内容提供商（记为 CP），为用户提供内容分发的平均流量为 λ。随机变量 X 表示给定时段内内容提供商收到的实际请求数。在任意时段，X 的分布事先已知而其实际值未知。内容提供商在分发内容时，可以直接对基础设施进行投资，使之具备单位时间内平均处理 I 个请求的能力。这时，其内容分发的收益为

$$U_{\mathrm{self}}(X)=V(X)-C(I)-c\cdot L(I,X) \tag{8.1}$$

式中，函数 V 是内容提供商响应所有 X 个请求的收益；函数 C 是设施的维护成本（如服务器、带宽、软件等），由于规模经济效益的原因函数 C 对 I 而言是一个凹函数[⊖]；函数 L 是被拒绝的请求数，它随着 X 的增加而增加，随着 I 增加而减小；c 是拒绝一个请求的成本。

⊖ 经济学中的凹函数与高等数学中的定义恰恰相反。—译者注。

函数 V 中涵盖了内容提供商在互联网运营（如在网上出售产品等）中的所有收入来源。内容提供商的相关设施成本可以建模为

$$C(I) = a_1 \cdot I - a_2 \cdot I^2,\ 对\ \forall I \leqslant a_1/2a_2$$

该式表明 I 和成本之间的函数是凹性的。约束条件 $I \leqslant a_1/2a_2$ 确保了网络设施的成本对于 I 而言总是单调上升的（注意对于 $I > a_1/2a_2$ 有 $C'(I) < 0$）。在该式中，a_1 取值大时表明网络设施成本高，而 a_2 取值大则说明规模经济效益明显。

由式（8.1），内容提供商分发内容得到的期望收益为

$$U_{\text{self}} = E[U_{\text{self}}(X)] = V - C(I) - c \cdot L(I) \tag{8.2}$$

式中，$L(I) = E[L(I, X)]$ 且 $V = E[V(X)]$。内容提供商选择一个合适的设施建设水平以最大化其期望收益。内容提供商的决策问题可以表示为

$$\max_{I} \{U_{\text{self}}(I)\}$$

用 I^* 表示网络设施建设的最优水平，相应的期望收益表示为 $U_{\text{self}}(I^*)$。

8.4.2　内容提供商通过 CDN 进行分发

对于内容提供商而言，另一种选择就是通过 CDN 来分发内容。这种情况下，内容提供商的收益为

$$U_{\text{CDN}}(X) = V(X) + \tau(N) \cdot X - C_o - P(X) \tag{8.3}$$

式中，函数 V 和 X 的定义如前文所述；函数 τ 指通过数量为 N 的一组 CDN 服务器实现更快的内容分发时由每个请求得到的收益；C_o 为内容分发的外包成本；函数 P 是内容提供商支付的 CDN 使用费。

内容提供商付出的外包成本包括内容修改成本或与 CDN 共享秘密内容的成本。前者与修改内容的成本相关，目的是加速 CDN 的内容分发。之所以存在共享秘密内容的成本，是因为内容提供商可能需要与 CDN 共享一些敏感信息，如消费者的相关记录、信用卡资料或病人的病历等。这种成本可能是某种形式的风险，也可能来自为确保安全而需要采取的额外步骤。共享秘密数据的成本一般因内容提供商而异，因为不同内容提供商处理的数据类型存在着内在的差异。

内容提供商并不清楚任一时段内有多少请求（X）出现，但是对于给定的任意价格函数 P（X），可以使用式（8.4）计算在使用 CDN 服务时的期望收益：

$$U_{\text{CDN}} = E[U_{\text{CDN}}(X)] = V + \tau(N)\lambda - C_o - E[P(X)] \tag{8.4}$$

如果满足条件 $U_{\text{CDN}} \geqslant U_{\text{self}}(I^*)$，则内容提供商选择购买 CDN 服务。将式（8.2）和式（8.4）代入这个条件可得，平均流量为 λ 和外包成本为 C_o 的内容提供商选择使用 CDN 服务的条件为

$$C_o \leqslant \tau(N)\lambda + C(I^*) + c \cdot L(I^*) - E[P(X)] \tag{8.5}$$

8.4.3　CDN 的收益函数

内容提供商选择自给的方式还是选择通过 CDN 的方式来分发内容，取决于哪种方式能够带来最大的期望收益。前面提过，CDN 并不清楚任何一个内容提供商的外包成本。因此，对于一个给定的 $P(X)$ 值，它并不能确定一个特定的内容提供商是否

会购买 CDN 服务。不过，CDN 知道 C_o在所有内容提供商上的分布，因此它可以计算内容提供商购买 CDN 服务的概率。由 $H(C_o)=C_o^W$，一个平均流量为 λ 的内容提供商购买 CDN 服务的概率为

$$\Pr(\text{购买} \mid \lambda) = (\tau(N)\lambda + C(I^*) + c \cdot L(I^*) - E[P(X)])^W \tag{8.6}$$

如果 $g(\lambda)$ 表示平均流量为 λ 的内容提供商的数量，那么购买 CDN 服务的内容提供商的期望数量为

$$Subs(\lambda) = g(\lambda)(\tau(N)\lambda + C(I^*) + c \cdot L(I^*) - E[P(X)])^W \tag{8.7}$$

对于请求 X 的一个实际值，任何一个购买 CDN 服务的内容提供商支付的费用为 $P(X)$。既然 X 不能预知，故 CDN 并不能通过一个价格函数 $P(X)$ 事先获悉自己的收益。然而，CDN 的期望收益可以通过式（8.8）计算。

$$\pi = \left\{\int_\lambda Subs(\lambda)\left(\int_X \Pr(X \mid \lambda) \cdot P(X)\,\mathrm{d}X\right)\mathrm{d}\lambda\right\} - \left\{b_1\left(\int_\lambda \lambda \cdot Subs(\lambda)\,\mathrm{d}\lambda\right) - b_2\left(\int_\lambda \lambda \cdot Subs(\lambda)\,\mathrm{d}\lambda\right)^2\right\} \tag{8.8}$$

在上面的表达式中，第一项代表了 CDN 的期望收益，其中，$\Pr(X \mid \lambda)$ 表示平均流量为 λ 的内容提供商收到 X 个请求的概率。因此，$\int_X \Pr(X \mid \lambda) \cdot P(X)\,\mathrm{d}X$ 代表了从一个平均流量为 λ 的内容提供商处获得的期望收益。利用式（8.8）对所有的内容提供商求和，可得 CDN 的总期望收益。第二项代表了 CDN 的成本，建模为 CDN 处理的平均流量的二次函数（由 $\int_\lambda \lambda \cdot Subs(\lambda)\,\mathrm{d}\lambda$ 给出）。这项成本包括与内容分发相关的成本，以及保持副本间内容一致性的成本。保持一致性的工作是一种记账机制，实现对与请求的路由和分发相关的信息进行收集和跟踪[20]。所有这些行为都具有规模效益，因此成本相对于流量应该是凹的，而这种凹性用二次函数来表示。注意，计算内容提供商和 CDN 提供商的成本时两者的参数是不同的（即 $a_1 \neq b_1$，$a_2 \neq b_2$)），这是因为 CDN 的成本还包括其他因素，如记账成本、内容一致性的维护成本以及内容分发成本。

CDN 的决策问题就是选择一个价格函数 $P(X)$，以实现其期望收益的最大化。

8.4.4 泊松分布和突发流量时的最优定价

本节通过仿真来确定 CDN 的最优定价策略。考虑一个由 1 000 个内容提供商构成的总体，内容提供商的平均抵达率服从区间［1000，8000］上的帕累托分布。对于给定的平均抵达率，流量可以通过如下分布生成。

1. 泊松分布

所有 1000 个内容提供商的流量服从泊松分布。

2. 突发流量

假设所有的内容提供商都有突发流量，并使用马尔科夫调制泊松过程（MMPP）对突发流量进行建模。MMPP 一般用于对 Web 服务器上的突发流量进行建模[1, 19]，它是一个双随机泊松过程，其中抵达率由一个 m 状态的马尔科夫过程给出。当马尔科夫链处于状态 j 时，请求的抵达服从抵达率为 λ_j的泊松过程。考虑一个简单的 2 状

态 MMPP，其抵达率分别为 λ_1 和 λ_2，相过程的极限状态概率为 $q=(q_1,q_2)$。单位时段内请求数的均值和方差分别记为 $\overline{\lambda}$ 和 Ψ。对一个流量突发进行建模时，假定以非零概率转移到状态 2，且 λ_2 具有一个很大的值。设 $\lambda_2=10\lambda_1$ 且 MMPP 的参数为（$q_1=0.9$，$q_2=0.1$），也就是说，在流量突发期间平均抵达率是正常情况下的 10 倍，同时系统负载平均而言上升了 10% 左右。通过改变 λ_1 的值可以模拟不同的均值 $\overline{\lambda}$。另外，当平均抵达率 $\overline{\lambda}$ 增加时，调整状态转移概率以维持恒定的突发状况（恒定值为 $\sqrt{\Psi/\overline{\lambda}}$）。

3. 混合流量

500 个内容提供商的流量服从泊松分布，另外 500 个服从 MMPP 分布。

仿真中的其他参数设置为：对于网络设施成本函数 $C(I)=a_1\cdot I-a_2\cdot I^2$，假设 $a_1=3.56$ 和 $a_2=0.000\ 043$，这些参数与现实中的网络设施成本大致相当。例如，在这种参数值设定下，每分钟响应 233 个请求的成本是每月 804 元。如果假设对请求的响应内容的平均大小为 100KB，那么就意味着响应数据在 3.10Mbit/s 的情况下成本为每月 804 元，这对于部分 T3 连接及维护低端服务器的成本来说是合理的。同样，每分钟响应 6 975 个请求的成本为 22 042 元，这大致上也就是 T3 连接的成本和相对应的维护服务器的成本，这些成本也与作者撰写本节时主机托管的成本大致相当。丢弃一个请求的成本 c 这里假设为 10 元，其依据是：假定 10% 的网站访问者购买产品/服务，每次购买的平均交易额为 100 元，而如果顾客的请求没有得到响应，则顾客就会离开网站。最后，假设内容提供商的外包成本服从区间 [0，30 000] 上的均匀分布。

基于上述设置，计算每一个内容提供商在自给情况下的最优网络设施配置和相关的期望收益。接着，计算在给定的 CDN 价格函数下，内容提供商采用 CDN 分发服务时的期望收益。对于 CDN 价格函数，这里仅采用二次函数，即 $P(X)=p_0\cdot X\pm p_1\cdot X^2$，并使用网格搜索的方法得到 p_0 和 p_1 的最优值。注意，上述价格函数对凸性和凹性的价格模式都可以实现建模。对于每一个内容提供商，根据相应的请求抵达分布（泊松或者 MMPP）为 X 生成 1000 个样本值。给定 p_0 和 p_1，可以计算对应于每一个 X 值的价格 $P(X)$，对 X 的 1000 个值取平均可以得到内容提供商的期望价格。给定这些参数，就可以计算出采用 CDN 分发服务时的期望收益。如果该收益大于自给情况下的收益，则内容提供商选择购买 CDN 服务。对内容提供商购买 CDN 服务所带来的期望收益求和，并减去 CDN 的成本（使用式（8.8）计算，其中 $b_1=3$，$b_2=0.000047$）就可以得到 CDN 的期望收益。通过 50 次仿真计算出最优价格，下面给出主要仿真结果。

1. 当流量服从泊松分布时最优定价需要使用总量折扣

当内容提供商的流量服从泊松分布时，CDN 的最优价格函数 $P^*(X)$ 为 $3.9X-0.000066X^2$。也就是说，在流量服从泊松分布时最优定价需要使用总量折扣。即使改变仿真的参数，最优价格仍保持凹性，而且折扣的大小随内容提供商设施成本中规模效益的增加（a_2 的增加）而增加。

2. 流量突发时最优价格较高

图 8.1 给出了流量在三种不同分布下的最优价格。可以看出，CDN 在流量突发

现象增多的情况下可以设定更高的价格，特别地，有 Price（MMPP）> Price（混合）> Price（泊松）。这是因为，存在突发流量时 CDN“避免请求丢失”这一针对内容提供商的价值主张得到了强化。因此，CDN 的收费价格会有较大提升。有意思的是，在混合情况下最优价格是凸性的，并存在量大多收费而不是量大优惠的情况，原因将在下面给出。

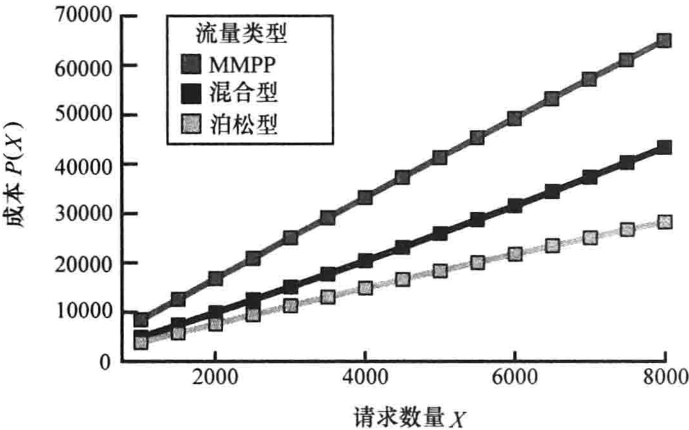


图 8.1 三种流量类型的最优价格函数

3. 在流量突发时传统的基于使用量的定价方式效率低下

考虑图 8.2 所示的总量折扣定价机制。内容提供商 CP_1 的平均抵达率为 $\overline{\lambda}$，不失一般性，假设 CP_1 有确定性的请求抵达过程。每个时段中，CP_1 收到 $\overline{\lambda}$ 个请求（图 8.2 中的 A 点）并向 CDN 支付期望价格 P_1。CP_2 具有相同的均值 $\overline{\lambda}$ 而方差较大。CP_2 以某些大概率收到请求（如图 8.2 中 B 点所示），但在余下的时间内，它收到大量请求（如图 8.2 中 C 点所示）。CP_2 支付的期望价格为 P_2，从图 8.2 中可以看出 P_2 明显低于 P_1。这一现象是由凹函数本身的特性造成的，与实际情况不符。方差较大的内容提供商反而从 CDN 获得更大的收益，这显然不是希望的结果，在理想情况下 CDN 应该向 CP_2 收取更高的价格。因此，即使自给方式下设施成本本身具有的凹性使价格函数趋于凹性，CDN 也应该选择凸函数作为定价函数。需要注意的是，只有当流量突发时各个内容提供商的参数存在差异时这种凸性才会出现，如果所有内容提供商在具有相同的平均抵达率的同时方差也相同，那么这种凸性就不会发生。

如果 CDN 使用一个凸的价格函数，那么平均抵达率高的内容提供商将会受罚。对于抵达率分别固定为 2λ 和 λ 的两个内容提供商，前者需要为使用 CDN 服务支付更高的额外费用。相比而言，抵达率为 2λ 的内容提供商在自给方式下能得到总量折扣，因此更倾向于用自给的方式来分发内容。因此，一个凸的价格函数会阻碍那些流量高和流量变化不大的内容提供商购买 CDN 服务。因此，在混合模式下最优价格函数的形状取决于流量突发在内容提供商之间的分布，也取决于内容提供商自身设施成本的总量折扣的大小。

以上分析间接地显示出在混合模式下传统的基于使用量的价格策略的低效性。该定价策略或者会惩罚低突发的内容提供商或者会惩罚高流量的内容提供商，这取决于

使用凸价格函数还是凹价格函数。因此，需要考虑另外一种基于某个使用量百分位数的定价策略。

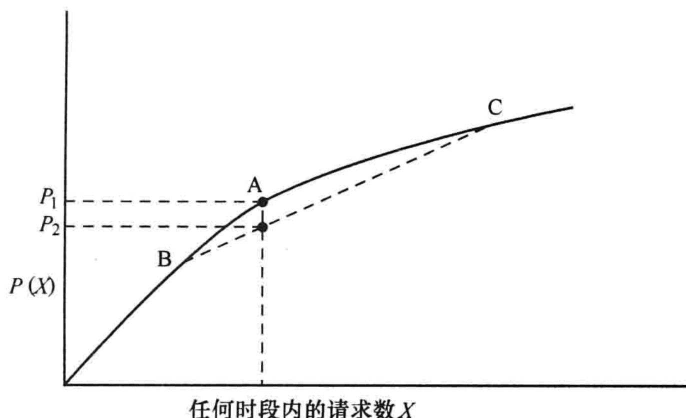


图 8.2　凹价格函数的期望价格

4. 基于百分位数的定价机制对于 CDN 而言更加有利可图

现在讨论基于使用量百分位数的定价机制。例如，CDN 监控某个时段内（如 1 个月）的请求率，并计算该时段内每个内容提供商的请求率的第 95 百分位数，对该内容提供商的定价就基于其带宽使用率的第 95 百分位数。令 Z 为请求率 X 的第 95 百分位数，与之前一样，为简化计算这里仅关注二次价格函数，设 $P(Z) = p_0 \cdot Z \pm p_1 \cdot Z^2$，并定量地计算基于百分位数的最优价格。图 8.3 中显示的是 CDN 在基于百分位数的定价机制和传统的基于使用量的定价机制下的期望收益。当各内容提供商的流量分布特性各异时，CDN 使用基于百分位数定价策略的收益要比传统的基于使用量的定价策略高。同时，流量服从单纯的泊松分布和 MMPP 时，基于百分位数的定价策略与传统的基于使用量的定价策略之间在收益上没有明显差别。这个现象并不奇怪，因为在这两种情况下一旦平均请求率固定下来，那么方差也随之固定[⊖]，从而一个基于均值的定价策略可以转化为基于百分位数的定价策略，反之亦然。在混合情况下，基于百分位数的定价机制允许 CDN 向内容提供商提供总量折扣，并同时向那些具有高突发流量的内容提供商设定更高的价格。

基于百分位数的收费策略也有不足之处：相对于传统方式而言收费机制更复杂，且缺乏标准化（例如，采样次数的选择会对收费产生影响）。这些问题已经使业内开始关注到底什么样的收费策略才是最合适的。由于这个原因，有些 CDN（如 SyncCast）采用了传统的基于使用量的收费方式，因为该方法的确简单。然而，本章的分析结果表明，当不同的内容提供商具有不同的流量突发特性时（现实中往往如此），基于百分位数的定价策略要比传统的基于使用量的定价策略更加有利可图。当各内容提供商的流量突发特性相似时，CDN 可以选择传统的基于使用量的定价机制来简化收费方式。

⊖ 对于泊松分布，方差等于均值。对于 MMPP 过程，方差等于流量突发（常量）与均值的乘积的二次方。——原书注

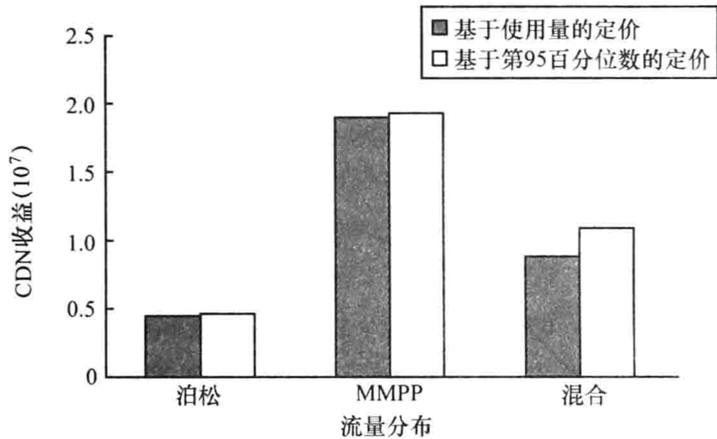


图 8.3 不同的定价策略和流量分布下 CDN 的收益

8.5 写给使用者

CDN 面临着一个复杂的定价问题。一方面，与传统的基于使用量的定价策略相比，基于百分位数的定价策略能产生明显的效益，而前者不能很好地监控流量的突发或峰值；另一方面，基于百分位数的定价策略给人的印象是针对流量峰值收费，因而受到了来自某些内容提供商和业界专家的反对。

尽管百分位数定价策略具有高收益性，这里仍然期望，内容提供商面对定价机制透明性的要求以及竞争压力的存在，可以最终促使百分位数定价策略的使用在未来几年有所下降。如果当前那些主导的 CDN 提供商将百分位数定价策略作为其首选，那么新进入该领域的 CDN 提供商就应该与之相区别，转而提供更加透明的定价方案。如果那些更透明的 CDN 定价方式被内容提供商采用，这将反过来迫使那些主导提供商也使用更加透明的定价方式。

另外一个可能性是更多的 CDN 将同时提供传统的基于使用量的定价策略和基于百分位数的定价策略，由内容提供商根据自身的偏好自行选择。事实上，若干 CDN 服务商已经开始提供这两种定价方式，并允许内容提供商自己选择。不过，这种情况对于 CDN 而言是否真的有好处尚不明朗。具有高突发流量的内容提供商愿意选择传统定价机制来避免因流量突发而带来的处罚；而那些具有低突发流量的内容提供商愿意选择百分位数计价方式，因为这种方案能够对具有高突发流量的内容提供商多收费而对具有低突发流量的内容提供商提供奖励。因此，同时提供这两种定价机制有可能会产生相反的结果：内容提供商会恰好选择 CDN 不希望它们选择的那种机制。因此，CDN 提供商最好能确定这样一种定价策略：既让内容提供商可以接受，又使自己能够有效地获得利润。

总之，CDN 提供商需要研究长远的解决方案。这些解决方案既包括培育内容提供商对百分位数定价方式的兴趣，也包括从行业内彻底取消百分位数定价方式。在这一点上，业界需要采取更加积极的态度。

8.6 未来研究方向

本章强调了传统的基于使用量的定价模式和基于百分位数的定价模式之间不存在一个简单的折中选择。一方面，基于百分位数的定价允许 CDN 在向一个流量大（流量变化不大）的内容提供商提供总量折扣的同时，对有着高突发流量的内容提供商提高收费。另一方面，基于百分位数的定价方式可能很复杂，从而可能会阻碍内容提供商购买 CDN 服务。因此，许多 CDN 提供商提供了这两种定价策略以便允许内容提供商自己选择。未来一个有吸引力的研究领域是这两种定价方法之间的交互，以及在同时提供这两种定价策略时如何确定最优的定价方式。

除此之外，很多对 CDN 定价的研究工作都集中在垄断这个假设条件下。未来一个令人感兴趣的研究方向是竞争对 CDN 定价策略的影响。虽然 Akamai 已经在这个产业内占据了统治地位，但近年来其竞争对手也取得了很大的发展，如 Limelight Networks 和 Level 3。各公司之间的竞争似乎已对占统治地位的公司的定价策略都产生了很大的影响[15]，因此研究竞争对 CDN 定价的影响将有助于推动这个行业的发展。

另一个趋势是 P2P 内容分发方式将成为 CDN 的一种替代物，同时也出现了很多 CDN 和 P2P 的混合体。P2P 和这些混合方案对 CDN 的影响现在还不清楚。最近 Johar 等人[11]研究了 P2P 媒体分发对一个纯 CDN 分发的收益的影响，并提出了很多令人感兴趣的结论。例如，他们发现，一个采用 P2P 分发方式的竞争者有时对 CDN 存在着积极的影响。但是，P2P 和 CDN 内容分发之间的相互关系仍需要进一步研究。

8.7 结论

内容分发网络的定价是一个复杂的问题。购买 CDN 服务的内容提供商在流量模式及其处理内容类型上存在着很大的差异，因而它们可能是高度异构的。同时，CDN 不得不使用单一的定价策略应对所有这些不同的流量类型。这也一直是该行业内存在很多争论的原因。

一些人对业内同时使用总量折扣和超额受罚的现象提出了质疑。本章强调的是内容分发中的规模经济效益推动了 CDN 定价中的总量折扣方式。但是，突发流量的存在又给超额受罚方式找到了存在的理由，一个解决办法是使用基于百分位数的收费。这种定价方式允许一个 CDN 向高流量的内容提供商提供总量折扣，同时向出现很高突发流量的内容提供商收取额外的费用。不过，业内的分析人士也对基于百分位数的定价方式提出了质疑，认为它类似于某种形式的峰值定价，但这种定价方式事实上的确有助于 CDN 获得更高的收益。

不过，还有很多问题尚未达成共识，如合适的 CDN 定价模型、竞争对定价和 CDN 市场效益的影响。对 CDN 的研究者和从业者来说，这将仍旧是一个相当吸引人的领域。

致谢

本章中的一些资料来自 HICSS'04[7]以及 Wharton School Working Paper[10]。

参考文献

[1] Anderson, M., Cao, J., Kihl, M., and Nyberg, C. Performance modeling of an Apache web server with bursty arrival traffic. In *Proc. of International Conference on Internet Computing (IC)*, June 2003.
[2] Cocchi, R., Shenker, S., Estrin, D., and Zhang, L. Pricing in computer networks: motivation, formulation and example. *IEEE/ACM Transactions on Networking*, vol 1, December 1993.
[3] Courcoubetis, C., Antoniadis, P. Market models for P2P content distribution. In *Proc. of First International Workshop on Agents and Peer-To-Peer Computing (AP2PC)*, 2002.
[4] Ercetin, O. and Tassiulas, L. Pricing strategies for differentiated services content delivery networks. *Computer Networks*, vol 49, no 6, pp 840–855, 19 December 2005.
[5] Gibbens, J. and Kelly, F.P. Resource pricing and the evolution of congestion control. *Automatica* 35, 1999.
[6] Gupta, A., Stahl, D.O., and Whinston, A. B. Priority pricing of integrated services networks. *Internet Economics*, eds Lee W. McKnight and Joseph P. Bailey, MIT Press, 1997.
[7] Hosanagar, K., Krishnan, R., Smith, M., Chuang, J. Pricing and service adoption of content delivery networks (CDNs). In *Proc. of the Hawaii International Conference on Systems and Sciences (HICSS)*, Hawaii, January 2004.
[8] Hosanagar, K., Krishnan, R., Chuang, J., and Choudhary. V. Pricing vertically differentiated web caching services. In *Proc. of the International Conference on Information Systems (ICIS)*, Barcelona, December 2002.
[9] Hosanagar, K., Krishnan, R., Chuang, J., and Choudhary, V. Pricing and resource allocation in caching services with multiple levels of quality of service. *Management Science*, vol 51, no 12, 2005.
[10] Hosanagar, K., Chuang, J., Krishnan, R., and Smith, M. Service adoption and pricing of content delivery network (CDN) services. *Management Science*, vol 54, no 09, 2008.
[11] Johar, M., Kumar, N., and Mookerjee, V. Analyzing the Impact of Peer-to-Peer Networks on the Market for Content Provision and Distribution. University of Texas, Dallas, Working Paper, 2007.
[12] Kaya, C., Dogan, K., and Mookerjee, V. An Economic and Operational Analysis of the Market for Content Distribution Services. In *Proc. of the International Conference on Information Systems*, Seattle, December 2003.
[13] Kelly, F. Charging and accounting for bursty connections. *Internet Economics*, eds Lee W. McKnight and Joseph P. Bailey, MIT Press, 1997.
[14] MacKie-Mason, J.K. and Varian, H.R. Pricing congestible network resources. *IEEE Journal of Selected Areas in Communications*, vol 13, no 7, pp 1141–149, September 1995.
[15] Malik, O. Akamai and the CDN Price Wars. GigaOM Blog, August, 2007.
[16] Mendelson, H. Pricing computer services: queuing effects. *Communications of the ACM*, vol 28, 1990.
[17] Mendelson, H., and Whang, S. Optimal incentive-compatible priority pricing for the M/M/1 queue. *Operations Research*, vol 38, 870–83, 1990.
[18] Rayburn, D. Content delivery pricing: understanding CDN overages." Streamingmedia Blog, October 2007.
[19] Scott, S. L. and Smyth, P. The Markov modulated Poisson process and Markov Poisson cascade with applications to web traffic modeling." In *Bayesian Statistics*, Oxford University Press, 2003.
[20] Vakali, A. and Pallis, G. Content delivery networks: status and trends. *IEEE Internet Computing* vol 7, no 6, pp 68–74, 2003.

第9章　CDN资源管理与分配的数学模型

Tolga Bektas 和 Iradj Ouveysi

9.1　引言

CDN的资源主要包括网络设施、在网络上分发的内容和分布在整个网络中包含若干资源的缓存服务器。为了获得具有成本效益的内容分发策略，CDN的资源需要有效地管理和分配。在提供CDN服务时，由于消费者的偏好开始成为必须考虑的一个关键因素，CDN提供商在设计其内容分发机制时也应该有一些服务质量（QoS）方面的考虑。

在CDN资源管理和分配问题上，数学建模是一个强大和有效的工具。本章的目的，是证明该领域中的大量问题都可以使用数学模型来精确地阐述，以及如何使用现有技术构造这些模型。为此，在9.2节将回顾一些近期的相关文献及相关工作，并提出相应的数学模型；在9.3节中介绍对这些模型的求解方法；在9.4节中提出一些适用于许多CDN结构的新模型；9.5节分析这些模型的效果；在9.6节中给出对相关使用者的建议；未来的研究方向和结论分别在9.7节和9.8节中给出。

9.2　相关工作

本节介绍与CDN资源管理和分配相关的数学模型，并对有关文献进行回顾。首先，定义一些本章将会用到的术语。内容（content）是指WWW中任何公开的可用信息，如网页、多媒体文件或文本文档等。内容中一个特定的项称为对象（object），如一个声音文件或一个文本文档。内容提供商发布供其他人访问的内容，而一个CDN提供商代替前者分发内容。这可能会存在一些例外，因为有的内容提供商自己完成内容的分发，因此本章假设内容提供商将这个任务外包给CDN提供商。客户端（client）是指为获取内容而发出请求的人或公司。CDN提供商将全部或部分内容保存在缓存服务器（caching server），缓存服务器部署在远程通信网络中，处理或转发客户端的请求。在随后的所有模型中，假设给定的是一个完全网络 $G=(V,E)$，其中 V 是节点的集合，$E=(\{i,j\}:i,j\in V)$ 是网络连接的集合。节点集 V 进一步分成三个非空子集合 I、J 和 S，其中 I 是客户端集合，J 是安装（或可以安装）缓存服务器[⊖]的节点集合，S 是源服务器的集合（当源服务器只有一个时 $S=\{0\}$）。本章提到的所有模型中所使用的符号参见表9.1。

⊖ 本文使用缓存服务器这一术语，而非代理服务器（proxy server），以避免混淆，因为代理服务器的概念原用于执行过滤和请求传递等，但随着网页变得快速和动态，这个概念已不再与实际相符。所以对于额外的服务器，称其为缓存寄存器（或简称为缓存）更合适一些。——原书注

9.2.1 基本问题

在 CDN 领域中，大部分复杂的数学模型都可用于构造一个具有成本效益的分发网，而设计这样一个分发网会面临三个基本问题。本节将简要地介绍一下这三个问题。

表 9.1 本章使用的符号汇总

集合	
I	客户端集合（$I \subset V$）
J	可以安装缓存服务器的节点集合（$J \subset V$）
S	源服务器的集合（$S \subset V$）
K	对象集合
参数	
b_k	对象 $k \in K$ 的大小
λ_i	客户端 $i \in I$ 的总请求率
h_{ij}	节点 $i \in I$ 发出的能被缓存服务器 $j \in J$ 服务的请求的比例
c_{ij}	节点 $i \in V$ 和 $j \in V$ 之间的距离，即跳数（成本）
f_{ij}	客户端 $i \in I$ 与缓存服务器 $j \in J$ 之间的流量数
d_{ik}	每单位时间客户端 $i \in I$ 请求对象 $k \in K$ 的请求率
f_j	对节点 $j \in J$ 上一个缓存服务器的操作成本
ψ_j	将一个对象放置在一个缓存服务器 $j \in J$ 的成本
β_j	缓存服务器 $j \in J$ 所需带宽的单位成本
δ_j	缓存服务器 $j \in J$ 所需处理能力的单位成本
C_j	缓存服务器 $j \in J$ 可用的计算能力单位的个数
s_j	缓存服务器 $j \in J$ 的存储能力
l_k	对象 $k \in K$ 所消耗的带宽
pw_k	对象 $k \in K$ 所消耗的处理能力
ρ_k	向客户端提供对象 $k \in K$ 带来的收益
L_{ij}	节点 $i \in V$ 和 $j \in V$ 之间的延时
Δ_d	延时的上界（可以从一个连接或一个对象或两者结合的角度定义）
p_{jk}	对象 $k \in K$ 存在于缓存服务器 $j \in K$ 的概率
变量	
$\theta_{jk} \in \{0, 1\}$	若对对象 $k \in K$ 的请求直接发给缓存服务器 $j \in J$ 则为 1，否则为 0
$x_{ij} \in \{0, 1\}$	若客户端 $i \in I$ 被分配给缓存服务器 $j \in J$ 则为 1，否则为 0
$x_{ijk} \in \{0, 1\}$	若客户端 $i \in I$ 请求的对象 $k \in K$ 在缓存服务器 $j \in J$ 上则为 1，否则为 0
$y_j \in \{0, 1\}$	若有一个缓存服务器在节点 $j \in J$ 上处于活动状态则为 1，否则为 0
$z_{jk} \in \{0, 1\}$	若对象 $k \in K$ 存在于缓存服务器 $j \in J$ 上则为 1，否则为 0
$u_k \in \{0, 1\}$	若对象 $k \in K$ 被复制（于任意缓存服务器 $j \in J$ 上）则为 1，否则为 0
$r_{ji}^k \geqslant 0$	访问被客户端 $i \in I$ 请求且发往缓存服务器 $j \in J$ 的对象 $k \in K$ 的比例

1. 缓存服务器的放置问题

对于一个现有的网络设施，缓存服务器的放置问题是指如何最优地在给定数量的

站点放置给定数量的服务器，使成本函数（总流量、所有客户端的平均延时、总分发成本）最小化[22]。Qiu 等人[26]提出了两个数学模型来解决缓存服务器的放置问题，即无容量限制的 p - 中值[2]和设施选址问题[11]。Krishnan[19]研究了如何透明地确定缓存的位置。该问题中考虑的目标函数很令人感兴趣，因为它考虑的情况是被请求的内容不在特定的缓存服务器中。这时，服务器 j 对客户端 i 进行服务的成本为

$$cost(i,j) = f_{ij}[h_{ij}c_{ij} + (1 - h_{ij})(c_{ij} + c_{js})] \tag{9.1}$$

式（9.1）所示的成本函数很好地描述了 CDN 的工作模式，并已被用于解决其他的问题[6][17]。

2. 对请求的路由

计算机网络中的路由是指在实现网络中总流量最小的前提下，将数据从一个或多个源节点传送到一个或多个目的地。参考文献［24］中对该问题进行了详细的回顾，并给出了组合优化应用的综述。另外，简单地说“对请求的路由”是将一个客户端的请求转发到能够对该请求进行服务的合适的缓存服务器的过程。对这个问题严格的定义是：对于访问服务器上某对象的一个请求，选择一个使成本函数最小的服务器来处理这个请求。这个问题的数学公式（尽管比较简单）可参见参考文献［14］。

3. 对象存储

前面提到的研究都是假设存储在源服务器中的内容已完全复制到缓存服务器中，这种情况下缓存服务器通常被称为副本或者镜像。不幸的是，如果对象非常大（如多媒体文件），那么缓存服务器有限的存储空间就只能允许复制部分的副本，在这种情况下任何缓存服务器都只能存储内容的一个子集。在有限的存储空间下，确定哪些对象应该被放置到缓存服务器上就是对象放置问题，该问题的数学模型可参见参考文献［18］。

9.2.2　综合问题

本节讨论 CDN 中一些更复杂的情况，即上面提到的那些问题可能会同时出现。首先研究一个没有源服务器的网络中的静态数据放置问题，目标是最小化总访问成本（客户端 $i \in I$ 从节点 $j \in J$ 获取对象 $k \in K$ 的成本为 $b_k d_{ik} c_{ij}$）。Baev[4]为这个问题提出的一个数学模型为

$$(\text{M1}) \quad \text{Minimize} \sum_{i \in I} \sum_{j \in I} \sum_{k \in K} b_k d_{ik} c_{ij} x_{ijk} \tag{9.2}$$

subject to

$$\sum_{j \in I} x_{ijk} = 1 \qquad \forall i \in I, k \in K \tag{9.3}$$

$$x_{ijk} \leqslant z_{jk} \qquad \forall i,j \in I, k \in K \tag{9.4}$$

$$\sum_{k \in K} b_k z_{jk} \leqslant s_j \qquad \forall i \in I \tag{9.5}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in I, k \in K \tag{9.6}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j \in I, k \in K \tag{9.7}$$

在模型 M1 中，z_{jk}是一个二值变量，当对象 $k \in K$ 存在于缓存服务器 $j \in J$ 中时

$z_{jk}=1$，否则 $z_{jk}=0$；二值变量 x_{ijk} 当客户端 $i \in I$ 请求的对象 $k \in K$ 在节点 $j \in J$ 有一个副本并得到处理时为1，否则为0。这个模型中，目标函数（9.2）表示对请求进行响应时所有节点和对象的总成本。注意，$J=I$ 意味着每个节点既充当客户端也作为一个潜在的缓存服务器。约束条件（9.3）表示每个节点的请求只能被转发给一个节点。约束条件（9.4）表示请求只能被分配给当前存储着被请求对象的那个节点。最后，约束条件（9.5）与每个节点 $j \in J$ 有限的存储空间 s_j 有关。

Laoutaris 等人[20]研究了相互联系的两个问题：对象向缓存服务器的放置和节点保存对象的存储空间规划，后者是确定一个给定的总存储容量以何种比例分配给网络中的每个节点，以实现所有客户端与所有被请求对象之间的平均距离最小。作者假设所有的对象都是单位大小。他们的另一个研究工作[21]提出了一个模型以解决 CDN 中存储空间的分配问题，该模型考虑的问题包括缓存服务器的安装位置、每个缓存服务器被分配的存储空间以及每个缓存服务器上应存储哪些对象等。这个模型定义在一个树状网络上，每个节点 i 都有一组记为 $a(i)$ 的父节点，一组记为 $l(i)$ 的叶节点。与参考文献［21］中的符号不同，使用上面定义的符号对该模型进行重写，得到模型 M2。

$$(\text{M2}) \quad \text{Maximize} \sum_{i \in I} \lambda_i \sum_{k \in K} d_{ik} \sum_{v \in a(i)} (c_{is} - c_{iv}) x_{ijk}$$

subject to

$$\sum_{v \in a(i)} x_{ijk} \leqslant 1 \qquad \forall i \in I, k \in K \tag{9.8}$$

$$\sum_{v \in l(i)} x_{ijk} \leqslant M z_{jk} \qquad \forall i \in I, k \in K \tag{9.9}$$

$$\sum_{j \in J} \sum_{k \in K} z_{jk} \leqslant D \tag{9.10}$$

$$x_{ijk} \in \{0,1\} \quad \forall i \in I, j \in J, k \in K \tag{9.11}$$

$$z_{jk} \in \{0,1\} \quad \forall j \in J, k \in K \tag{9.12}$$

可以发现，这个模型与 Baev 等人提出的模型 M1 非常相似，不同之处在于目标函数，这里的目标函数是使对象放置在缓存服务器上所节省的成本最大化。约束条件（9.8）和（9.9）与客户端向缓存服务器的分配有关，其中的 M 是一个充分大的数。约束条件（9.10）保证所有缓存服务器中分配给对象的存储空间的总和不能超过所有缓存服务器的总可用空间，后者记做 $D = \sum_{j \in J} s_j$。既然 Laoutaris 等人[21]假设所有的对象都是单位大小，那么节点中可用于存储对象的空间大小就等于节点上放置对象的个数。

Nguyen 等人[23]考虑了在共享网络设施上配置 CDN 的问题，并提出了一个联合配置和对象复制的模型，实现存储、请求处理和启动等方面的总成本最小化。下面将使用本章定义的符号来介绍这个模型，并引入如下参数：一个对象能够以单位放置成本 ψ_j 和单位带宽成本 β_j 放置在每个缓存服务器上；一个缓存服务器的单位处理能力成本记为 δ_j，且可用的处理能力为 C_j 个单位；对每个对象 k 的处理消耗 l_k 个单位的带宽和 c_k 个单位的处理能力；单位时间内服务提供商可以从每个对象 k 上获利 ρ_k；两个节

点 i 和 j 之间的延时记为 L_{ij}，其上界是 Δ_d；二值决策变量 u_k 表示一个对象是否在任何缓存服务器上被复制；变量 r_{ji}^k 表示客户端 i 对对象 k 的请求应该分配给服务器 j。

$$\text{(M3)}\quad \text{Maximize}\quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} r_{ji}^k\rho_k - \sum_{j\in J}\sum_{k\in K}\Psi_j b_k z_{jk} - \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} r_{ji}^k(\beta_j l_k + \delta_j c_k) - \sum_{j\in J} f_j y_j \tag{9.13}$$

subject to

$$\sum_{i\in I} r_{ji}^k p w_k \leqslant C_j y_j \quad \forall j \in J \tag{9.14}$$

$$\sum_{j\in J} r_{ji}^k p w_k = u_k d_{ik} \quad \forall i \in I, k \in K \tag{9.15}$$

$$r_{ji}^k (L_{ij} - \Delta_d) \leqslant 0 \quad \forall i \in I, j \in J, k \in K \tag{9.16}$$

$$r_{ji}^k \leqslant d_{ik} z_{jk} \quad \forall i \in I, j \in J, k \in K \tag{9.17}$$

$$y_j \in \{0,1\} \quad \forall j \in J \tag{9.18}$$

$$u_k \in \{0,1\} \quad \forall k \in K \tag{9.19}$$

$$z_{jk} \in \{0,1\} \quad \forall j \in J, k \in K \tag{9.20}$$

模型 M3 的目标函数是最大化服务提供商的收益，其中收益等于总收入（即式（9.13）的第一个项）减去总成本，总成本来自存储、带宽、CPU 以及网站建设等方面。约束条件（9.14）是对每个服务器的容量约束；约束条件（9.15）保证所有的请求都得到响应。约束条件（9.16）确保所有的请求都在允许的延时内得到响应。最后，约束条件（9.17）的作用是保证只有当缓存服务器中存在被请求对象时，请求才能被该缓存服务器响应。

Bektaş 等人研究了服务器位置、对象放置和请求路由的问题。在参考文献［9］中，他们提出了一个新的模型，该模型考虑到多对象，并结合一个合适的、阿尔伯特非线性、类似于式（9.1）的目标函数，对标准的设施选址模型进行了扩展，使之适用于 CDN。他们提出的这个整数规划模型为

$$\text{(M4)}\quad \text{Minimize}\quad \sum_{j\in J} f_j y_j + \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}\left[b_k d_{ik} c_{ij} z_{jk} x_{ij} + b_k d_{ik}(1 - z_{jk})(c_{j0} + c_{ij}) x_{ij} \right] \tag{9.21}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \quad \forall i \in I \tag{9.22}$$

$$x_{ij} \leqslant y_j \quad \forall i \in I, j \in J \tag{9.23}$$

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j y_j \quad \forall j \in J \tag{9.24}$$

$$y_j \in \{0,1\} \quad \forall j \in J \tag{9.25}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I, j \in J \tag{9.26}$$

$$z_{jk} \in \{0,1\} \quad \forall j \in J, k \in K \tag{9.27}$$

模型 M4 的目标函数是将式（9.1）推广到多客户端、多服务器和多对象的情形。模型中的第一项表示构建缓存服务器的总成本；第二项由两部分组成：第一部分对应于缓存服务器响应用户请求的成本，而第二部分是当缓存服务器中没有被请求对象

时，它从源服务器获取该对象的额外成本。约束条件（9.22）保证每个客户端的请求都被分配到唯一的缓存服务器；约束条件（9.23）表示只有当该缓存服务器处于可用状态时才可以接受分配；约束条件（9.24）确保向每个缓存服务器放置对象时不会超出其自身的存储容量。

Bektaş 等人[8]随后又从实际操作层面出发研究了这个问题，他们忽略了缓存服务器的放置问题，但同时施加了一个 QoS 约束，以限制对象点对点传输的延时，由此提出的模型为

(M5)　Minimize $\sum_{i\in I}\sum_{j\in J}\sum_{k\in K}[b_k d_{ik} c_{ij} z_{jk} x_{ij} + b_k d_{ik}(1 - z_{jk})(c_{j0} + c_{ij})x_{ij}]$

subject to

$$\sum_{j\in J} x_{ij} = 1 \qquad \forall i \in I \tag{9.28}$$

$$\sum_{j\in J} L_{ij} x_{ij} z_{jk} + \sum_{j\in J}(L_{ij} + L_{j0})x_{ij}(1 - z_{jk}) \leqslant \Delta_d \qquad \forall i \in I, j \in J, k \in K \tag{9.29}$$

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j y_j \qquad \forall j \in J \tag{9.30}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{9.31}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K \tag{9.32}$$

模型 M5 与模型 M4 结构相似，只是如上所述忽略了缓存服务器的放置问题，并纳入了 QoS 约束，即约束条件（9.29）。式（9.29）可以写为更简单的形式[8]：

对于 $\forall i \in I, j \in J, k \in \mathscr{D}_{ij}$，有

$$x_{ij} \leqslant z_{jk}$$

其中，对于每一对（i，j）有

$$\mathscr{D}_{ij} = \{k \in K \mid L_{ij} \leqslant \Delta_d \text{ 且} (L_{ij} + L_{j0}) > \Delta_d\}$$

上式中的 $\mathscr{D}_{ij}$ 是所有对象的一个子集，对于其中的每个对象，缓存服务器 j 将该对象发送到发起请求的客户端 i 所需的时间 L_{ij} 不超过允许的延时 Δ_d。但是，如果缓存服务器 j 没有该对象，则会增加一个从源服务器取对象的额外时间 L_{j0}，且总时间会超出 Δ_d。所以，由于 QoS 的限制将不允许缓存服务器从源服务器中读取该对象。

以上所有模型都是假设 CDN 只有一个源服务器，也就是说 $|S| = 1$，这与大多数实际情况相吻合。但在某些情况下，内容提供商也许会（可能在同一个站点）部署多个源服务器，其原因一般是为了增加系统的稳定性或扩大系统容量。参考文献[6]中提出了一个模型，解决多个源服务器情况下缓存服务器放置、请求路由以及对象存储等问题，该模型为

$$\text{(M6)} \quad \text{Minimize} \sum_{j\in J} f_j y_j + \sum_{i\in I}\sum_{j\in J}\sum_{s\in S}\sum_{k\in K}[b_k d_{ik} c_{ij} z_{jk} x_{ij} + b_k d_{ik}(1 - z_{jk})(c_{ij} + c_{js} t_{js})x_{ij}] \tag{9.33}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \qquad \forall i \in I \tag{9.34}$$

$$x_{ij} \leqslant y_j \qquad \forall i \in I, j \in J \tag{9.35}$$

$$\sum_{k \in K} b_k z_{jk} \leqslant s_j y_j \qquad \forall j \in J \tag{9.36}$$

$$\sum_{s \in S} t_{js} = 1 \qquad \forall j \in J \tag{9.37}$$

$$y_j \in \{0,1\} \qquad \forall j \in J \tag{9.38}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{9.39}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K \tag{9.40}$$

$$t_{js} \in \{0,1\} \qquad \forall j \in J, s \in S \tag{9.41}$$

模型 M6 将模型 M4 推广到多个源服务器，并引入了一个二值变量 t_{js}：如果缓存服务器 $j \in J$ 分配给一个源服务器 $s \in S$ 则 $t_{js} = 1$，否则 $t_{js} = 0$。相对于模型 M4，对其目标函数进行了扩展，以考虑到所有可用的源服务器。另外，增加了约束条件（9.37），以确保每一个缓存服务器应该被分配给唯一的源服务器，当缓存服务器上不存在被请求的对象时，它将把该请求转发至相应的源服务器。

9.3　求解算法

上述问题以及相关数学模型的求解算法有两类。第一类是精确算法，可以产生最优解，但代价是需要相当长的计算时间；第二类是启发式算法，计算量一般相对较小，但不幸的是这类算法通常不能保证得到最优解。在目前可用的大量第一类算法中，本节着眼于两种基于分解的算法，因为它们可以将原问题分解成小规模的、更易解决的多个子问题。

9.3.1　Benders 分解

Benders 分解[10]允许将问题分解成两个子问题。为了介绍 Benders 分解，假设一个给定模型为

$$(\mathscr{P}) \quad \text{最小化 } \boldsymbol{cx} + \boldsymbol{fy} \text{ 当}: \boldsymbol{Ax} + \boldsymbol{By} = \boldsymbol{d}, \boldsymbol{x} \in X, \boldsymbol{y} \in Y \tag{9.42}$$

其中 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 是变量构成的列向量，$\boldsymbol{c}$ 和 $\boldsymbol{f}$ 是成本系数构成的行向量，$\boldsymbol{A}$ 和 $\boldsymbol{B}$ 是约束系数矩阵，$\boldsymbol{d}$ 是等号右侧各值构成的列向量。变量 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 分别定义在非空集合 X 和 Y（假设前者是连续的集合，后者是整数集合）。

首先将问题（$\mathscr{P}$）重写为下面的形式：

$$(\mathscr{P}_1) \quad \min_{\tilde{y} \in Y} \{ f\tilde{y} + \min_{x \in X} \{ \boldsymbol{cx}: \boldsymbol{AX} = \boldsymbol{d} - \boldsymbol{B}\tilde{\boldsymbol{y}} \} \} \tag{9.43}$$

其中 $\boldsymbol{y}$ 预设初值为 $\boldsymbol{y} = \tilde{\boldsymbol{y}}$。既然问题 $\mathscr{P}_1$ 中内侧的那个仅与变量 $\boldsymbol{x}$ 有关的最小化问题（记做 $\mathscr{P}$）是线性且连续的，所以在 $\mathscr{P}$ 的每个约束条件中可以用 $\boldsymbol{x}$ 的对偶变量 $\boldsymbol{w}$ 对 $\boldsymbol{x}$ 进行替换，得到式（9.44）：

$$\min_{\tilde{y} \in Y} \{ \boldsymbol{f}\tilde{\boldsymbol{y}} + \max_{\boldsymbol{w}} \{ \boldsymbol{w}(\boldsymbol{d} - \boldsymbol{B}\tilde{\boldsymbol{y}}): \boldsymbol{wA} \leqslant \boldsymbol{c} \} \} \tag{9.44}$$

假设 $\mathscr{P}$ 的对偶问题的可行域是非空的（否则就意味着原问题无解或无界），原问

题 ($\mathscr{P}$) 可以重写为

$$\text{Minimize} \quad z + fy \tag{9.45}$$

subject to

$$z \geqslant \tau^r(d - By) \quad \tau^r \in \gamma \tag{9.46}$$

$$\zeta^u \ (d - By) \ \leqslant 0 \quad \zeta^u \in \Psi \tag{9.47}$$

$$y \in Y \tag{9.48}$$

重写后的问题称为主问题。对偶问题的可行解空间的极点和极射线分别用 γ 和 Ψ 表示。约束条件 (9.46) 是针对 $\mathscr{P}$ 的对偶问题的可行域的每一个极点，而约束条件 (9.47) 针对的是与对偶问题的不可行解相对应的每一个极射线。

参考文献 [8] 指出，模型 M5 有一个特殊的结构，使其可以应用 Benders 分解。这里引入辅助的线性变量 φ_{ijk}，对应于模型 M5 的目标函数中的乘积项 $x_{ij}z_{jk}$，下面将以模型 M5 的线性情况为例证明这一点，细节参见参考文献 [8]。

$$\text{Minimize} \quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}[b_k\lambda_{ik}(c_{ij} + c_{j0})x_{ij} - b_k\lambda_{ik}c_{j0}\varphi_{ijk}] \tag{9.49}$$

subject to

$$\sum_{j\in J} x_{ij} = 1, \qquad \forall i \in I \tag{9.50}$$

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j \qquad \forall j \in J \tag{9.51}$$

$$\varphi_{ijk} - x_{ij} \leqslant 0 \qquad \forall i \in I, j \in J, k \in K \tag{9.52}$$

$$\varphi_{ijk} - z_{jk} \leqslant 0 \qquad \forall i \in I, j \in J, k \in K \tag{9.53}$$

$$x_{ij} - z_{jk} \leqslant 0 \qquad \forall i \in I, j \in J, k \in \mathscr{D}_{ij} \tag{9.54}$$

$$x_{ij} \geqslant 0 \qquad \forall i \in I, j \in J \tag{9.55}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K \tag{9.56}$$

$$\varphi_{ijk} \in \{0,1\} \qquad \forall i \in I, j \in J, k \in K \tag{9.57}$$

将上述线性模型中的对象位置变量固定取值为 $z_{jk} = \bar{z}_{jk}$，得到的子问题能够进一步以每个客户端 $i \in I$ 为单位分解为更小的问题：

$$\text{Minimize} \quad \sum_{j\in J}\sum_{k\in K}[b_k\lambda_{ik}(c_{ij} + c_{j0})x_{ij} - b_k\lambda_{ik}c_{j0}\varphi_{ijk}] \tag{9.58}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \tag{9.59}$$

$$\varphi_{ijk} - x_{ij} \leqslant 0 \qquad \forall j \in J, k \in K \tag{9.60}$$

$$\varphi_{ijk} \leqslant z^*_{jk} \qquad \forall j \in J, k \notin \mathscr{D}_{ij} \tag{9.61}$$

$$x_{ij} \leqslant z^*_{jk} \qquad \forall j \in J, k \in \mathscr{D}_{ij} \tag{9.62}$$

$$x_{ij} \geqslant 0 \qquad \forall j \in J$$

尽管每个子问题仍是整数规划问题，但具有完整性的特性。这个特点使对变量 x_{ij} 的约束可以放宽，从而求解其对偶。令约束条件 (9.59) ~ (9.62) 的对偶变量分别为 α_i 、θ_{ijk}、ω_{ijk} 和 ζ_{ijk}，构造主问题为

$$\text{Minimize} \quad \sum_{i\in I}\xi_i \tag{9.63}$$

subject to

$$\xi_i + \sum_{j\in J}\sum_{k\in K} z_{jk}(\widetilde{\omega}_{ijk} + \widetilde{\zeta}_{ijk}) \geq \widetilde{\alpha}_i \qquad (\alpha,\theta,\omega,\zeta) \in \mathscr{P}_i^{\mathscr{D}} \tag{9.64}$$

$$\widetilde{\alpha}_i - \sum_{j\in J}\sum_{k\in K} z_{jk}(\widetilde{\omega}_{ijk} + \widetilde{\zeta}_{ijk}) \leq 0, \qquad (\alpha,\theta,\omega,\zeta) \in \mathscr{W}_i^{\mathscr{D}} \tag{9.65}$$

$$\sum_{k\in K} b_k z_{jk} \leq s_j \qquad \forall j \in J$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K$$

其中，式（9.64）为使用对偶问题最优解的对应系数时的最优约束，计算公式为

$$\widetilde{\alpha}_i = \min_{j\in\mathscr{F}_i\cup\mathscr{H}_i}\left\{\sum_{k\in K} b_k\lambda_{ik}(c_{ij}+c_{j0}) - \sum_{k\in K:z_{jk}^*=1} b_k\lambda_{ik}c_{j0}\right\}$$

$$\widetilde{\omega}_{ijk} = \begin{cases} b_k\lambda_{ik}c_{j0}, & \text{如果 } z_{jk}^* = 0 \\ 0, & \end{cases}$$

$$\sum_{k\in K:z_{jk}^*=0} \widetilde{\zeta}_{ijk} = \widetilde{\alpha}_i + \sum_{k\in K}\widetilde{\theta}_{ijk} - \sum_{k\in K} b_k\lambda_{ik}(c_{ij}+c_{j0}) \quad \forall j \in J$$

式中，$\mathscr{F}_i = \{j \in J \mid \mathscr{L}_{ij} = \mathscr{R}_{ij} = \phi\}$ 且 $\mathscr{H}_i = \{j \in J \mid z_{jk}^* = 1, \forall k \in \mathscr{L}_{ij}\}$。约束条件（9.65）针对的是对偶问题的不可行解所对应的每个极射线。鉴于主问题中最优性和不可行性约束条件的数量，故对这个问题的求解是不现实的。因此，需要求助于这样一个策略：从只包含有限约束条件的受限主问题开始，新的约束条件迭代地加入这个受限问题，直到求出最优解。参考文献［8］中介绍了这个方法的细节，以及各种提高算法效率的精调方法，如使用帕累托优化约束及约束的去除。

9.3.2 拉格朗日松弛和分解

拉格朗日松弛可以将一个模型中的部分约束条件以拉格朗日形式对偶化（松弛），以变换为一个易解的问题。考虑前面提到的问题 $\mathscr{P}$，假设 $\boldsymbol{Ax}+\boldsymbol{By}=\boldsymbol{d}$ 是那些使模型“复杂”的约束条件，引入一个拉格朗日乘数向量 μ，可以将这些约束条件对偶化为

$$(\mathscr{P}_\mu) \quad \text{Minimize } \boldsymbol{cx}+\boldsymbol{fy}+\mu(\boldsymbol{Ax}+\boldsymbol{By}-\boldsymbol{d})$$

subject to $\quad \boldsymbol{x} \in \boldsymbol{X}, \boldsymbol{y} \in \boldsymbol{Y}$

引入松弛向量后，可以很容易地将 $\mathscr{P}_\mu$ 直接分解成两个子问题，其中只涉及变量 $\boldsymbol{x}$ 的子问题为

$$\min_{x\in X}(\boldsymbol{c}+\mu\boldsymbol{A})\boldsymbol{x}$$

另一个只涉及变量 y 的子问题为

$$\min_{y\in Y}(\boldsymbol{f}+\mu\boldsymbol{B})\boldsymbol{y}$$

对于 μ 的任意给定值，$\mathscr{P}_\mu$ 的解都是其最优解的一个下界。为了寻找这个下界，需要求解分段线性凹优化问题 $\max_\mu \mathscr{P}_\mu$，这个问题称为拉格朗日对偶问题，可以通过不可微优化技术求解。

拉格朗日松弛法可以解决前面提到的一些问题，如 Qiu 等人[26]提出的服务器放置问题、Nguyen 等人[23]提出的覆盖分布网络的配置问题以及 Bektaş 等人[8]提到的对

象放置和请求路由问题。本节介绍这个技术如何应用于参考文献［8］中提出的一个整数线性规划模型，不过使用的是另一种类型的松弛方法。下面给出的模型是引入辅助变量 v_{jk} 后模型 M5 的一个线性化变形。

$$\text{Minimize} \quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} b_k\lambda_{ik}(c_{ij}+c_{j0})x_{ij} - \sum_{j\in J}\sum_{k\in K} v_{jk} \tag{9.66}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \qquad \forall i \in I \tag{9.67}$$

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j \qquad \forall j \in J \tag{9.68}$$

$$v_{jk} - Mz_{jk} \leqslant 0 \qquad \forall j \in J, k \in K \tag{9.69}$$

$$v_{jk} - \sum_{i\in I} b_k\lambda_{ik}c_{j0}x_{ij} \leqslant 0 \qquad \forall j \in J, k \in K \tag{9.70}$$

$$x_{jk} - z_{jk} \leqslant 0 \qquad \forall i \in I, j \in J, k \in \mathscr{Z}_{ij} \tag{9.71}$$

$$x_{ij} \geqslant 0 \qquad \forall i \in I, j \in J \tag{9.72}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K \tag{9.73}$$

$$v_{jk} \geqslant 0 \qquad \forall j \in J, k \in K \tag{9.74}$$

使用拉格朗日乘数 σ_i、π_{jk} 和 η_{jk} 分别对约束条件（9.67）、（9.69）和（9.70）进行对偶化，可得松弛后的问题 $\mathscr{R}$：

$$(\mathscr{R}) \quad \text{Minimize} \quad \sum_{i\in I}\sum_{j\in J}\Big(\sum_{k\in K}\{b_k d_{ik}[c_{ij}+c_{j0}(1-\eta_{jk})]\} - \sigma_i\Big)x_{ij}$$

$$+ \sum_{j\in J}\sum_{k\in K}(\pi_{jk}+\eta_{jk}-1)v_{jk} - M\sum_{j\in J}\sum_{k\in K}\pi_{jk}z_{jk} - \sum_{i\in I}\sigma_i \tag{9.75}$$

subject to

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j \qquad \forall j \in J \tag{9.76}$$

$$x_{ij} - z_{jk} \leqslant 0 \qquad \forall i \in I, j \in J, k \in \mathscr{L}_{ij} \tag{9.77}$$

$$x_{ij} \geqslant 0 \qquad \forall i \in I, j \in J \tag{9.78}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K \tag{9.79}$$

$$v_{jk} \geqslant 0 \qquad \forall j \in J, k \in K \tag{9.80}$$

问题 $\mathscr{R}$ 可以分解成两个子问题：一个只涉及变量 $\boldsymbol{x}$ 和 $\boldsymbol{z}$，另一个只涉及变量 $\boldsymbol{v}$。后者可以通过确定 v_{jk} 来求解：如果 $\pi_{jk}+\eta_{jk}-1$ 非负，则 $v_{jk}=0$，否则 $v_{jk}=1$。至于前一个子问题，注意到变量 $\boldsymbol{x}$ 只在约束条件（9.77）和（9.78）中出现，因此可以这样确定变量 $\boldsymbol{x}$：如果 $\sum_{k\in K}\{b_k d_{ik}[c_{ij}+c_{j0}(1-\eta_{jk})]\}-\sigma_i$ 非负，则 $x_{ij}=0$，否则 $x_{ij}=\hat{z}_{jk}$，其中 $\hat{z}_{jk}$ 是下面这个问题的解：

$$\text{Maximize} \quad \sum_{j\in J}\sum_{k\in K}\pi_{jk}z_{jk}$$

subject to

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j \qquad \forall j \in J$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K$$

而这个问题又可以进一步分解为一系列二值背包问题，对应于每个 $j \in J$ 都是一

个背包问题，可以使用动态规划方法在 $\mathcal{O}(|K| s_j)$ 的时间内解决。

9.3.3 启发式算法

与精确算法相反，启发式算法是一种快速和灵活的求解方法，可以适用于精确算法无能为力的场合，如 CDN 提供商的问题。贪婪启发式算法是一种启发式算法，它对于给定的问题，可以在每一步迭代中从所有可能的替代方案中选择最好的那个，即能够最大限度地降低成本的那个。但是，这类方法不能保证得到的结果具有全局最优性，因此大多数情况下只能获得局部最优解。不过，这类启发式算法的优势在于处理问题时的计算速度以及解决非常大的问题时的灵活性，这就解释了贪婪启发式算法为什么能在 CDN 中得到广泛应用[18,21,26,27,31]。近似算法是另一种启发式算法，可以保证得到的解与最优解之间的差距在一个常数比例之内，这种方法在 CDN 中的应用参见参考文献 [4，20]。模拟退火算法和禁忌搜索算法是更复杂的启发式算法，称为超启发式算法，它们采用特定的机制防止对解的搜索陷入局部最优。参考文献 [7] 中提出了模拟退火算法的一个两级实现，用于解决对象放置和请求路由问题。禁忌搜索算法在相同问题上的一个应用可参见参考文献 [15]。

表 9.2 给出了目前 CDN 在资源分配和管理中使用的各种模型和解决方案，如服务器放置（SP）、请求路由（RR）、对象放置（OP）以及内容分发（CD），同时包括一些相关的文献。

表 9.2　现有模型及求解方法的分类

服务器放置	请求路由	对象放置	内容分发	参考文献	求解方法
x				[22]	动态规划
x				[26]	拉格朗日松弛，贪婪启发式
x				[27]	贪婪启发式
x				[5]	贪婪启发式
x				[17]	动态规划
	x			[14]	整数规划模型
		x		[13]	贪婪启发式
		x		[18]	贪婪启发式
		x	x	[20]	精确算法、近似算法
		x	x	[21]	贪婪启发式
x	x			[1]	启发式算法
x		x		[29]	动态规划
x		x		[30]	启发式算法
	x	x		[4]	近似算法
	x	x		[3]	启发式算法
	x	x		[28]	解析式算法、启发式算法
	x	x		[7]	模拟退火
	x	x		[8]	Benders 分解、拉格朗日松弛
	x	x		[15]	禁忌搜索
x	x	x		[23]	拉格朗日松弛
x	x	x		[9]	Bender 分解、贪婪启发式

9.4 其他 CDN 架构使用的新模型

前面章节介绍的大多数模型都基于各种不同的 CDN 架构。为了建模的方便，对这些模型使用了严格的约束，这使它们都有各自的局限性。本节将提出一种新的模型，用于更普遍的情形，从数学建模的角度来说这项工作尚属首创。为了方便对模型的描述，这里使用一个小规模的例子。该例中，网络结构由单个源服务器（$|S|=1$）、3 个缓存服务器（$J=\{1, 2, 3\}$）以及 10 个客户端（$I=\{1, 2, \cdots, 10\}$）组成，网络拓扑如图 9.1 所示。在这个例子中，假设网络中分布着 5 个对象（$K=\{1, 2, \cdots, 5\}$），它们的大小（单位为 GB）分别为 $b_1=94$、$b_2=75$、$b_3=96$、$b_4=61$ 及 $b_5=82$。缓存服务器的容量分别设置为 $s_1=156$、$s_2=162$ 及 $s_3=85$（单位同样为 GB），其大小占所有对象总大小的比例从 20% 到 40% 不等。对于任意的 $i\in I$ 和 $j\in J$，节点之间的距离 c_{ij} 在 1 ~ 5 之间随机取值（距离矩阵参见附录中的表 9.4），而每个缓存服务器和源服务器之间的距离设定为 $c_{1,0}=20$、$c_{2,0}=15$ 及 $c_{3,0}=18$。为简单起见，假设每个客户端对每个对象的请求率都相同（即对所有的 $i\in I$ 和 $k\in K$ 有 $d_{ik}=1$）。

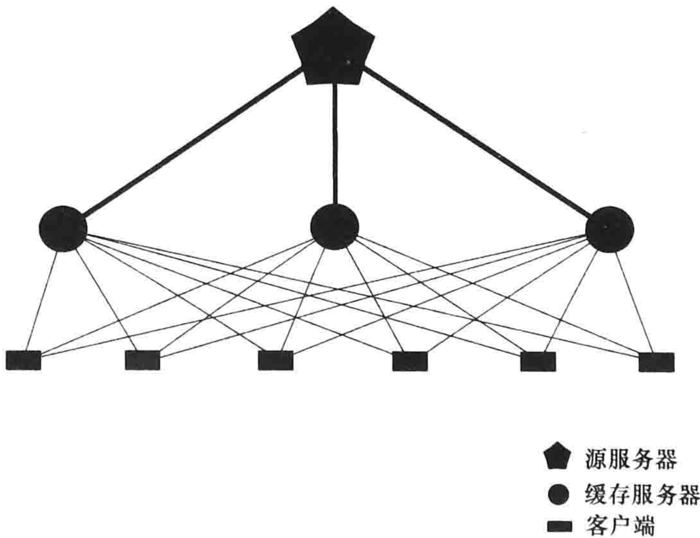


图 9.1 样例网络的拓扑架构

使用模型 M4 求解样例问题，并将其作为基准。对于本节所有的模型，都使用最新的非线性整数规划求解器 BONMIN[㊀] 在 NEOS 服务器在线完成求解[㊁]。M4 的最优解如图 9.2 所示，其中客户端到服务器的分配用粗连线表示，对象在每个缓存服务器的放置也可从图中看到，求解的总成本为 31229。

㊀ 参见 https：//projects. coin - or. org/Bonmin。——原书注

㊁ 参见 neos. mcs. anl. gov/neos/solvers/minco：Bonmin/AMPL. html。——原书注

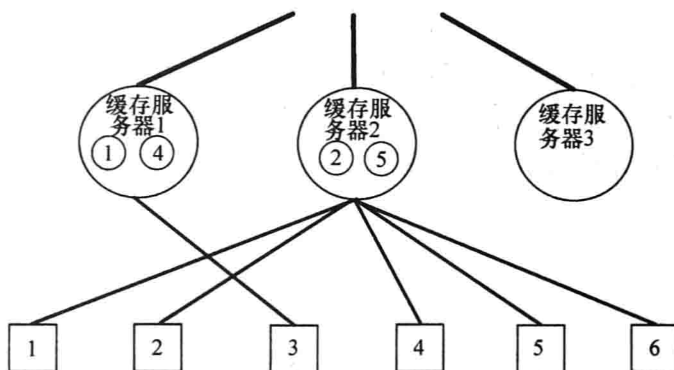


图 9.2　使用模型 M4 得到的样例问题的解

9.4.1　从多服务器获取对象

在前面提出的模型中通常假定 CDN 的结构为：一个客户端 $i \in I$ 被分配给单个缓存服务器 $j \in J$（以下称主服务器），从主服务器可获取所请求的对象。当被请求的对象不在主服务器 $j \in J$ 中时，主服务器将请求转发给源服务器以获取对象。当缓存服务器数量很多，且从其他缓存服务器获取对象的管理成本比较高时，这样的策略可能比较恰当。但是，当存在着大量具有相似请求率的对象，且缓存服务器的存储空间有限（意味着将有大量请求转发给源服务器）时，这么做可能并非总是一个可行的选择。为了解决这个问题，可以采用另一种策略，即只有当对象在任何缓存服务器上都不存在时，才将客户端对该对象的请求转发给源服务器（参见 Datta 等人[14] 的工作）。这意味着，每个客户端可以从其他的缓存服务器中获得对象。但这时就会出现第二个问题：如果所有的缓存服务器中都没有被请求对象时该如何应对？为了解决此问题，首先做出以下约束：客户端的请求只能通过其主服务器转发给源服务器（本节随后会讨论放宽此约束时的情况）。此时模型为

$$(\text{M7})\quad \text{Minimize}\quad \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} b_k d_{ik} c_{ij} x_{ijk} + \sum_{i\in I}\sum_{k\in K}\Big\{\sum_{j\in J}\Big[b_k d_{ik}(c_{ij}+c_{j0})x_{ij}\Big(1-\sum_{t\in J}x_{itk}\Big)\Big]\Big\} \tag{9.81}$$

subject to

$$\sum_{j\in J} x_{ij} = 1 \quad \forall i \in I \tag{9.82}$$

$$\sum_{j\in J} x_{ijk} \leqslant 1 \quad \forall i \in I, k \in K \tag{9.83}$$

$$\sum_{k\in K} b_k z_{jk} \leqslant s_j \quad \forall j \in J \tag{9.84}$$

$$\sum_{j\in J} x_{ijk} \geqslant z_{jk} \quad \forall i \in I, j \in J, k \in K \tag{9.85}$$

$$x_{ijk} \leqslant z_{jk} \quad \forall i \in I, j \in J, k \in K \tag{9.86}$$

$$x_{ij} + z_{jk} - x_{ijk} \leqslant 1 \quad \forall i \in I, j \in J, k \in K \tag{9.87}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I, j \in J \tag{9.88}$$

$$x_{ijk} \in \{0,1\} \quad \forall i \in I, j \in J, k \in K \tag{9.89}$$

$$z_{jk} \in \{0,1\} \quad \forall j \in J, k \in K \tag{9.90}$$

在模型 M7 中，目标函数由两部分组成。第一部分表示直接从（多个）缓存服务器对客户端进行服务的成本，第二部分表示只有当其他的缓存服务器上都没有被请求的对象时，主服务器才把请求转发给源服务器，即对于所有的 $j \in J$ 有 $z_{jk}=0$，也就是说，下式成立：

$$(1 - \sum_{i \in J} x_{itk}) = 1$$

约束条件（9.82）表示每个客户端向其主服务器的分配；约束条件（9.83）限定每个客户端至多只从一个缓存服务器获取每个对象；约束条件（9.86）确保请求只能由拥有被请求对象的单个缓存服务器进行服务；对每个缓存服务器存储容量的限制体现在约束条件（9.87）中，对于任何请求该约束条件给出了在拥有对象的情况下（即如果 $x_{ij}=z_{jk}=1$，则 $x_{ijk}=1$），主服务器处理请求的优先级。模型 M7 的解如图 9.3 所示，其最优解为 14001，与 M4 模型相比降低了大约 55%。

在图 9.3 中，客户端向主服务器的分配由粗连线表示，而路由到其他服务器的请求由细连线表示（即客户端 1、3、4 和 5 分配给缓存服务器 2，但它们从缓存服务器 1 中获得对象 1 和 4，这是因为它们的主服务器中没有这些对象）。在这个例子中，任何对于对象 2 的请求不得不最终由源服务器进行服务，因为没有一个缓存服务器拥有该对象。客户端 1、3、4 和 5 通过它们的主服务器（2 号）获得对象 2，而客户端 2 和 6 通过它们的主服务器（3 号）获取该对象。

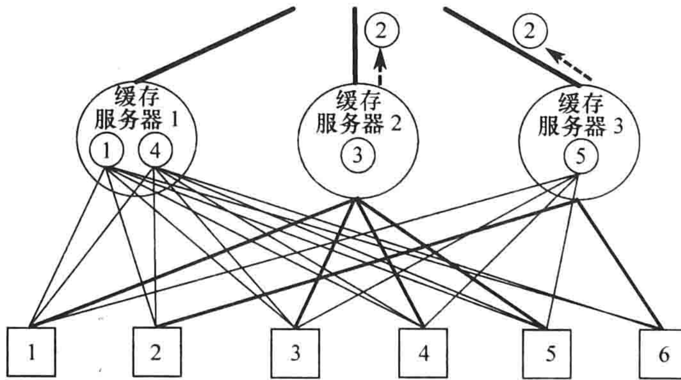


图 9.3　使用模型 M7 得到的样例问题的解

为了使分发策略具有更多的灵活性，下面提出另一种模型：允许客户端的对象请求可以被网络中的任何缓存服务器转发（并服务）。在这个模型中，定义了一个新的二值变量 v_{ijk}，如果源服务器经由缓存服务器 j 将对象 k 送达客户端 i，v_{ijk} 为 1，否则为 0。

（M8）　Minimize $\sum_{i \in I} \sum_{j \in J} \sum_{k \in K} [b_k d_{ik} c_{ij} x_{ijk} + b_k d_{ik}(c_{ij} + c_{j0}) v_{ijk}]$

subject to

$$\sum_{j \in J} (x_{ijk} + v_{ijk}) = 1 \quad \forall i \in I, k \in K \tag{9.91}$$

$$\sum_{k \in K} b_k z_{jk} \leqslant s_j \quad \forall j \in J \tag{9.92}$$

$$x_{ijk} \leqslant z_{jk} \quad \forall i \in I, j \in J, k \in K \tag{9.93}$$

$$v_{ijk} \leqslant 1 - z_{jk} \qquad \forall i \in I, j \in J, k \in K \tag{9.94}$$

$$x_{ijk}, v_{ijk} \in \{0,1\} \qquad \forall i \in I, j \in J, k \in K \tag{9.95}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K \tag{9.96}$$

模型 M8 的目标函数由两部分组成，第一部分表示对象从缓存服务器传送到客户端的总成本，第二部分表示从源服务器获取对象的成本。约束条件（9.91）确保客户端请求的任何对象，要么直接从一个缓存服务器得到，要么经由某个缓存服务器从源服务器得到；约束条件（9.92）对缓存服务器的容量进行限制；约束条件（9.93）表示，只有当一个缓存服务器拥有被请求对象时，客户端才能得到该缓存服务器的服务；约束条件（9.94）确保，只要至少有一个缓存服务器 $j \in J$ 拥有被请求对象，则该对象就不能由源服务器提供。模型 M8 的解如图 9.4 所示，最优解为 13940，比 M7 得到的解还要低。图 9.4 中显示的结果体现了使用该模型后分发过程具有的灵活性，任何客户端都能从（或经由）任何服务器接收到任何对象。例如，客户端 1 从缓存服务器 1 中接收到对象 1，从缓存服务器 2 中得到对象 3 和 4，从缓存服务器 3 中得到对象 5，通过缓存服务器 2 得到对象 2（该缓存服务器将请求转发给了源服务器）。

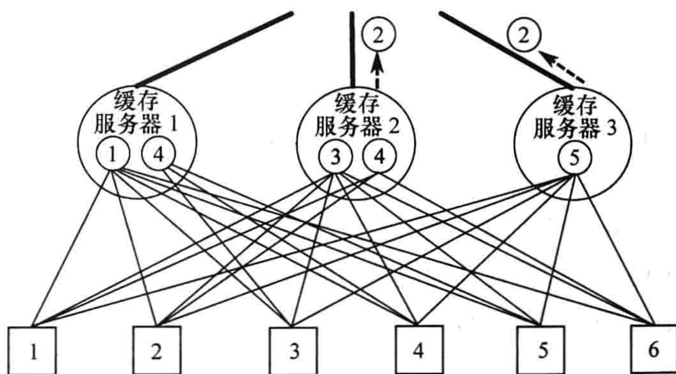


图 9.4　使用模型 M8 得到的样例问题的解

9.4.2　CDN 的生存力设计

远程通信网络的生存力（Survivability）是指其在一条链路或一个服务器失效时的运行能力。对于前者，目前已有很多相关文献进行了研究（如参考文献［25］）。在设计 CDN 时可以采纳其研究成果，在客户端和服务器之间建立备用连接，以便当主连接失效时激活该备用连接。而对于后者，则与本章的工作非常相关，因为上面提出的模型都是基于这样的假设，即每个客户端都连到单个缓存服务器，并通过该缓存服务器得到相应的服务。当主服务器失效时，客户端需要立即由另一个缓存服务器服务（即使被请求的对象并不在该服务器中，因为缓存服务器只是充当了一个通往源服务器的中转）。所以，一个 CDN 若想具有生存力，就需要这样的设计：每个客户端都应分配给一个备用的（或随时待命的）服务器，这样当主服务器失效时，请求应该转向备用服务器。基于上述分析，根据生存力的要求对模型 M7 进行扩展，得到的新模型为

$$\text{(M9)} \quad \text{Minimize} \quad \sum_{j \in J} f_j y_j + \sum_{i \in I}\sum_{j \in J}\sum_{k \in K}\left[b_k d_{ik} c_{ij} z_{jk} x_{ij}^{p} + b_k d_{ik}(1 - z_{jk})(c_{jS} + c_{ij}) x_{ij}^{p}\right]$$
$$\gamma \sum_{i \in I}\sum_{j \in J}\sum_{k \in K}\left[b_k d_{ik} c_{ij} z_{jk} x_{ij}^{p} + b_k d_{ik}(1 - z_{jk})(c_{jS} + c_{ij}) x_{ij}^{b}\right] \tag{9.97}$$

subject to

$$\sum_{j \in J} x_{ij}^{p} = 1 \qquad \forall i \in I \tag{9.98}$$

$$\sum_{j \in J} x_{ij}^{b} = 1 \qquad \forall i \in I \tag{9.99}$$

$$x_{ij}^{p} + x_{ij}^{b} \leqslant y_j \qquad \forall i \in I, j \in J \tag{9.100}$$

$$\sum_{k \in K} b_k z_{jk} \leqslant s_j y_j \qquad \forall j \in J \tag{9.101}$$

$$y_j \in \{0,1\} \qquad \forall j \in J \tag{9.102}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J \tag{9.103}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in J, k \in K \tag{9.104}$$

在模型M9中，x_{ij}^{p}是一个二值变量：当服务器j是客户端i的主服务器时，其值为1，否则为0。X_{ij}^{b}是另一个二值变量，当缓存服务器j是客户端i的备用服务器时，值为1，否则为0。目标函数（9.97）中的前两项与M4类似，第三项表示当发生故障时向客户端提供备用服务的成本。因为发生故障的频率不一定很高，所以该成本不会经常发生，可以使用参数$0 \leqslant \gamma \leqslant 1$调节备用服务器成本对CDN设计的影响。这样，当$\gamma = 0$时，CDN提供商就不用考虑向其客户提供备用服务器的成本；当$\gamma = 1$时，总成本包括提供备用服务器的成本（即使服务也许根本不会使用）。在该模型中，约束条件（9.98）与主服务器的分配有关，而约束条件（9.99）确保每个客户端也同时被分配到一个备用服务器。约束条件（9.101）是对处于激活状态的缓存服务器施加的容量限制。使用模型M9对样例求解，得到的结果如图9.5所示，当$\gamma = 0.7$时最优解为55561.7，当$\gamma = 0.3$时最优解则为41657.3。

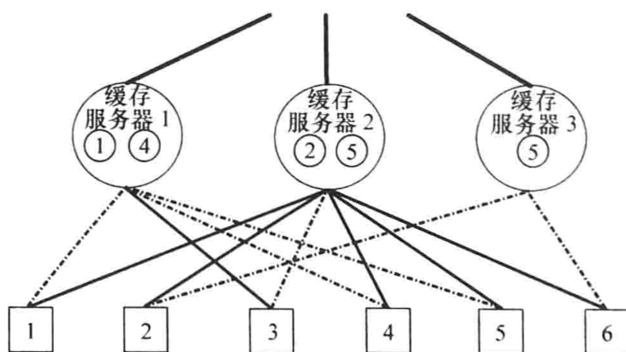


图9.5　使用模型M9得到的样例问题的解

如图9.5所示，主服务器的分配用粗连线表示，备用服务器的分配用虚连线表示。对于客户端1来说，缓存服务器2作为其主服务器，当它失效时客户端1立刻被路由到缓存服务器1。

9.5　性能结果

为了方便读者查看新模型M7～M9的计算性能，本章给出的是运行在一组样例下的部分计算实验的结果。这些样例的产生方式与参考文献［9］中的方式相同，都是基于一个具有3个缓存服务器和10个客户端的网络。被分发对象的数量在20～90之间变化，间隔为10。此例中，对象请求率并不均匀，而是根据Zipf－like分布产生（参见参考文献［12，32］），其形式为$P_K(i) = \Omega i^{-\alpha}$，其中$\Omega = (\sum_{j=1}^{k} j^{-\alpha})^{-1}$是归一化常量，分布的参数设为$\alpha = 0.733$。

实验结果在图 9.6 中给出，图中显示了使用模型 M4、M7、M8 和 M9 得到的解（运行 2 次，γ 分别为 0.3 和 0.7）。如图所示，模型 M7 和 M8 的性能很相似，都比 M4 的结果要好。另外，由于引入了生存力，M9 的解具有很高的成本。求解这些模型所花费的时间如表 9.3 所示。这些值意味着，即使问题的规模很小，M7 和 M8 的求解也被证明是非常困难的。实际上，以 $|K|=80$ 和 $|K|=90$ 为例，这些模型的最优解也无法在 1h 内计算完成（图 9.6 中的数值是在这个时间限度内可能获得的最好结果）。尽管表 9.3 中显示的数值意味着其他模型的求解时间应该会随着样例规模的增大而变长，但它们对于这些样例则更易于求解。

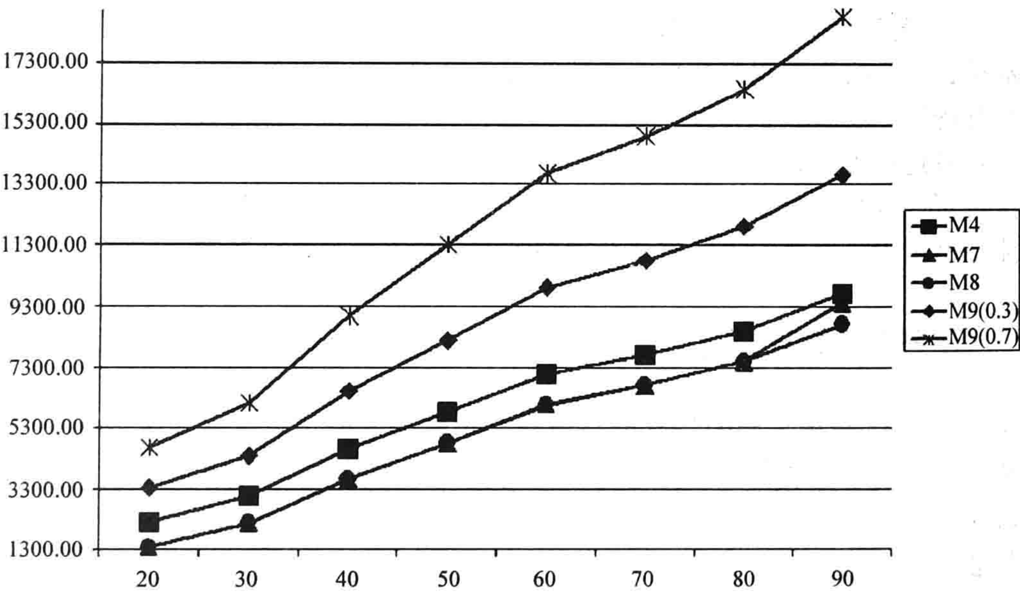


图 9.6　各个模型在样例下的成本比较

表 9.3　各个模型在样例下的求解时间（s）

$\|I\|$	$\|J\|$	$\|K\|$	M4	M7	M8	M9（0.3）	M9（0.7）
3	10	20	0.02	230.48	161.29	0.05	0.23
3	10	30	0.05	256.54	679.54	0.09	0.42
3	10	40	0.31	118.17	88.29	1.19	0.50
3	10	50	0.34	161.58	418.15	0.29	0.21
3	10	60	0.16	1764.42	1867.74	1.01	0.77
3	10	70	0.13	384.59	395.48	0.13	0.79
3	10	80	0.27	3600.00	3600.00	1.80	1.94
3	10	90	0.45	3600.00	3600.00	1.23	3.90

9.6　写给使用者

很明显，对于一个像 CDN 这样的动态和活跃的环境，大多数应用都需要快速、可扩展的方法。在所有这类方法中，启发式方法是一种普遍使用的方法，贪婪启发式

方法[19]、拓扑结构已知的启发式方法[16]或者热点算法[26]已广泛用于缓存服务器定位问题。不过，虽然可以说这种启发式方法优于另一种启发式方法，但对于这些方法得出的解的质量，目前并没有一个评估的指标。本章的目的是强调使用数学模型特别是精确算法来解决 CDN 问题的重要性，并使读者认识到使用这些方法是有好处的。的确，数学模型可以用做基准，从解的质量角度评估各种启发式方法，这自然有助于在实际中选择合适的启发式方法。例如，Laoutaris 等人[21]就采用这样的方式，通过与一个精确算法进行比较，对一个贪婪算法（及其变种）的性能进行评估。

对于 CDN 在实际中出现的各种问题，利用数学建模技术也可以加深对这些问题的理解，以决定采用何种缓解措施。例如，Nguyen 等人[23]在数学建模的基础上，使用了一个基于拉格朗日方法的求解算法来评估对象聚类对 CDN 提供商总收入的影响。

最后，数学模型可以很容易地增减约束条件或更改问题的参数，这种灵活性方便了对各种场景的分析，也有助于决策者选择合理的方案。例如，一个 CDN 提供商可能希望在特定的参数设置下，对不同的路由策略或对象放置策略进行评估。尽管启发式方法也可以实现这一点，但与使用数学模型获得的结果相比，启发式方法获得的结果可能不够精确，因为启发式方法获得的解的质量并不总是已知的。

9.7 未来研究方向

未来在 CDN 建模方面的研究可分为两个主要的方向：提出新的模型和对现有算法的改进。对前者而言，本章提出的新模型表明，确实有一些场景尚未被建模，这也意味着实际中极有可能出现更复杂的场景，并由此可能发展出其他的一些模型。在这方面的建议是，将生存力问题或缓存服务器的放置决策引入模型 M7 或 M8，或者将 QoS 约束（如参考文献［8］中的那些约束）引入模型 M7 ~ M9。这种尝试无疑会导致更复杂的模型，而这些模型很可能会表现为非线性整数规划的形式。

本章通过定量实验证明，即使对于小规模的问题而言，M7 或 M8 等模型的求解也是非常困难的，这就要求设计出新的精确算法，以高效地求解这些复杂的模型。在精确算法的问题上，基于分解并结合线性化策略的方法，对于非线性模型来说是一个有希望的研究方向。然而，这些精确算法很可能无法解决较大规模的问题，这就需要研究快速和高扩展性的启发式方法。启发式方法和精确算法的研究应该同时进行，互为补充。事实证明，在某些问题上[7,9,15]这种策略非常有助于开发出更好的方法。

9.8 结论

本章简要介绍了在 CDN 资源（网络、缓存服务器和对象）管理和分配上的基本问题，并介绍了一些已有的数学模型，这些模型用于在一般框架下解决上述问题。另外，本章提供一些例子及相应的论述，介绍了如何使用一些精确算法和启发式方法来求解相关的模型以解决这些问题。针对各种尚未深入研究的情况，本章也提供了一些新的数学模型，如设计一个具有较高生存力的 CDN。

数学模型是一个强大的工具，可以解决 CDN 提供商面临的问题，并可以对问题

的本质获得更深入的理解。正如在前面提到的，数学模型可以提供一个一般性框架，通过在这个框架上设计高效的精确算法来方便了问题的求解。对于精确算法而言，那些基于分解的算法最有可能成功地求解 CDN 问题。在评估启发式方法的质量时，精确算法也是至关重要的，特别是评估广泛用于很多 CDN 问题中的贪婪启发式方法。

致谢

本章中的一些资料来自 Computers & Operations Research Journal 杂志[8,9]。

附录

样例问题的距离矩阵（可解释为两个节点 $i \in I$ 和 $j \in J$ 之间的跳数）如表 9.4。

表 9.4　样例问题的距离矩阵

c_{ij}	$j=1$	$j=2$	$j=3$
$i=1$	1	1	3
$i=2$	5	4	1
$i=3$	1	5	5
$i=4$	1	4	5
$i=5$	1	3	2
$i=6$	5	5	1

参 考 文 献

[1] Almeida, J., Eager, D., Vernon, M., Wright, S.: Minimizing delivery cost in scalable streaming content distribution systems. IEEE Transactions on Multimedia **6**, 356–365 (2004)

[2] Avella, P., Sassano, A., Vasil'ev, I.: Computational study of large-scale p-median problems. Mathematical Programming **109**, 89–114 (2007)

[3] Backx, P., Lambrecht, T., Dhoedt, B., DeTurck, F., Demeester, P.: Optimizing content distribution through adaptive distributed caching. Computer Communications **28**, 640–653 (2005)

[4] Baev, I., Rajaraman, R.: Approximation algorithms for data placement in arbitrary networks. In: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 661–670 (2001)

[5] Bassali, H., Kamath, K., Hosamani, R., Gao, L.: Hierarchy-aware algorithms for CDN proxy placement in the Internet. Computer Communications **26**, 251–263 (2003)

[6] Bektaş, T.: Discrete location models for content distribution. Unpublished PhD Dissertation, Bilkent University, Ankara, Turkey (2005)

[7] Bektas, T., Cordeau, J.F., Erkut, E., Laporte, G.: A two-level simulated annealing algorithm for efficient dissemination of electronic content. Journal of the Operational Research Society **35**, 3860–3884 (2008)

[8] Bektaş, T., Cordeau, J.F., Erkut, E., Laporte, G.: Exact algorithms for the joint object placement and request routing problem in content distribution networks. Computers & Operations Research (2008). In press

[9] Bektaş, T., Oğuz, O., Ouveysi, I.: Designing cost-effective content distribution networks. Computers & Operations Research **34**, 2436–2449 (2007)

[10] Benders, J.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik **4**, 238–252 (1962)

[11] Berman, O., Krass, D.: An improved IP formulation for the uncapacitated facility location problem: Capitalizing on objective function structure. Annals of Operations Research **136**, 21–34 (2005)
[12] Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web caching and Zipf-like distributions: evidence and implications. In: Proceedings of IEEE INFOCOM'99, Vol. 1, pp. 126–134. New York (1999)
[13] Cidon, I., Kutten, S., Soffer, R.: Optimal allocation of electronic content. Computer Networks **40**, 205–218 (2002)
[14] Datta, A., Dutta, K., Thomas, H., VanderMeer, D.: World Wide Wait: a study of Internet scalability and cache-based approaches to alleviate it. Management Science **49**, 1425–1444 (2003)
[15] Dubuc, G., Bektaş, T., Cordeau, J.F., Laporte, G.: Une heuristique de recherche avec tabous pour la conception de réseaux de distribution de contenu électronique INFOR **45**, 175–185 (2007)
[16] Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., Zhang, L.: On the placement of Internet instrumentation. In: Proceedings of IEEE INFOCOM'00, pp. 295–304 (2000)
[17] Jia, X., Li, D., Hu, X., Wu, W., Du, D.: Placement of web-server proxies with consideration of read and update operations on the Internet. The Computer Journal **46**(4), 378–390 (2003)
[18] Kangasharju, J., Roberts, J., Ross, K.: Object replication strategies in content distribution networks. Computer Communications **25**, 376–383 (2002)
[19] Krishnan, P., Raz, D., Shavitt, Y.: The cache location problem. IEEE/ACM Transactions on Networking **8**, 568–582 (2000)
[20] Laoutaris, N., Zissimopoulos, V., Stavrakakis, I.: Joint object placement and node dimensioning for Internet content distribution. Information Processing Letters **89**, 273–279 (2004)
[21] Laoutaris, N., Zissimopoulos, V., Stavrakakis, I.: On the optimization of storage capacity allocation for content distribution. Computer Networks **47**, 409–428 (2005)
[22] Li, B., Golin, M., Italiano, G., Deng, X., Sohraby, K.: On the optimal placement of web proxies in the Internet. In: Proceedings of IEEE INFOCOM'99, Vol. 3, pp. 1282–1290. New York (1999)
[23] Nguyen, T., Safaei, F., Boustead, P., Chou, C.: Provisioning overlay distribution networks. Computer Networks **49**, 103–118 (2005)
[24] Oliveira, C., Pardalos, P.: A survey of combinatorial optimization problems in multicast routing. Computers & Operations Research **32**, 1953–1981 (2005)
[25] Ouveysi, I., Wirth, A., Yeh, A., Oguz, O.: Large scale linear programs and heuristics for the design of survivable telecommunication networks. Annals of Operations Research **124**, 285–293 (2003)
[26] Qiu, L., Padmanabhan, V., Voelker, G.: On the placement of web server replicas. In: Proceedings of IEEE INFOCOM'01, Vol. 3, pp. 1587–1596 (2001)
[27] Radoslavov, P., Govindan, R., Estrin, D.: Topology informed Internet replica placement. Computer Communications **25**, 384–392 (2002)
[28] Wauters, T., Coppens, J., De Turck, F., Dhoedt, B., Demeester, P.: Replica placement in ring based content delivery networks. Computer Communications **29**, 3313–3326 (2006)
[29] Xu, J., Li, B., Lee, D.: Placement problems for transparent data replication proxy services. IEEE Journal on Selected Areas in Communications **20**, 1383–1398 (2002)
[30] Xuanping, Z., Weidong, W., Xiaopeng, T., Yonghu, Z.: Data Replication at Web Proxies in Content Distribution Network, *Lecture Notes in Computer Science*, Vol. 2642, pp. 560–569. Springer-Verlag, Xian (2003)
[31] Yang, M., Fei, Z.: A model for replica placement in content distribution networks for multimedia applications. In: Proceedings of IEEE International Conference on Communications (ICC '03), Vol. 1, pp. 557 –561 (2003)
[32] Zipf, G.: Human Behavior and the Principle of Least-Effort. Addison-Wesley, Cambridge, MA (1949)

第 10 章　全球覆盖路由的性能和可用性效益

Hariharan S. Rahul，Mangesh Kasbekar，Ramesh K. Sitaraman 和 Arthur W. Berger

10.1　引言

人类历史经历了很多转折点，在这些转折点处技术革新以不可逆转的方式从根本上改变了人类生活的各个方面。毫无疑问，现在我们又面临着一个新的转折点：互联网革命。然而，互联网要想成为下一个革命性的全球通信媒介，就必须解决五个技术挑战：极高的可用性、高性能、“无限”的可扩展性、完善的安全性以及可承受的成本。

互联网最初的设计目标并不是作为一个具有高可用性的通信媒介，所以它目前没有提供多少我们所需要的东西也就不足为奇。因此，需要一些大的科技创新将互联网的潜力转变成实际的好处。CDN 覆盖在传统互联网之上，作为实现上述目标的一项新技术，已显示出巨大的前景。

10.1.1　CDN 架构回顾

为了方便读者理解本章的内容，首先简要回顾一下商用 CDN 的体系架构及其发展演变过程，更详尽的内容可参见第 1 章。

CDN 出现之前，内容提供商的典型做法是在一个数据中心配置一个中央服务器集群和若干个流服务器，向全球的终端用户（也称为客户端）提供内容服务。然而，这种方案在可用性、性能以及扩展性方面存在严重的不足。内容从源服务器进入互联网时会存在所谓的“第一公里”瓶颈问题，而当内容跨越多个远距离网络和对等点时又会遇到“中间公里”瓶颈（middle - mile bottleneck）问题。在“第一公里”内，数据中心本身就是一个单点失效。在数据中心处发生的任何一个连接性问题（如一个超负载或出问题的交换机），都会导致可用性的降低甚至彻底瘫痪。而在中间公里处，当内容远距离传输时经过潜在拥堵的对等点时，往返延时和损耗的上升会使可用性和性能大大降低。另外，对于瞬时拥塞问题也没有防御机制，除非数据中心一开始就获得足够高的配置。

对托管内容的数据中心采用多宿（multihoming）技术可以缓解传统托管方式的一些缺陷[3]。也就是说，数据中心通过多条链路与多个网络提供商连接，并使用一定的路由策略来控制不同链路上的流量。另一个互补方案也可以缓解集中托管面临的问题，即在位于不同地域和网络的多个数据中心内，保留同一份内容的镜像。上述两种方法可以改善第一公里处的一些可用性问题，缓解因单一数据中心或网络的失效导致的整个网站的瘫痪，但中间公里处的性能下降和可扩展性问题仍有待解决。另外，由

于要有效地管理多条链路和多个数据中心，运营成本和复杂度也会增加。不仅如此，因为在发生故障时部分链路和数据中心必须能够承担整个系统的负载，所以网络和服务器资源往往需要过度地配置。对可用性和性能的更高要求就需要配置更大的服务器集群和更多的镜像，这意味着更高昂的设施建设成本。因此，具有庞大共享的分布式平台的 CDN 就显示出其吸引力。

从上 1 章的讨论中可知，CDN 是一种服务器的分布式网络，它覆盖在互联网之上，以高性能、高可靠性、高扩展性和低成本的方式向客户端提供内容。一个高度简化的 CDN 架构包括五个主要部分，如图 10.1 所示。

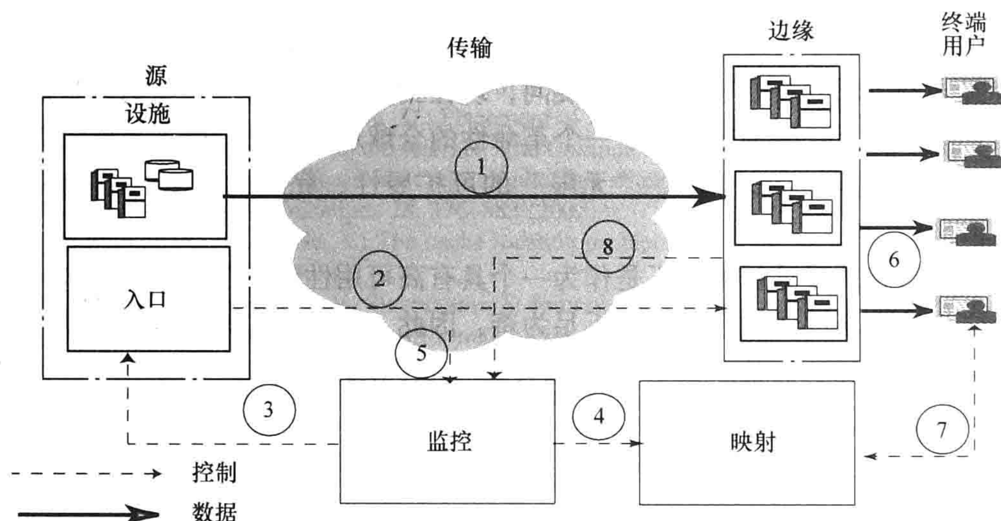


图 10.1　一个 CDN 的高级架构

1. 边缘系统

该系统包括网络、流或边缘服务器，边缘服务器位于离客户端很近的互联网“边缘”地带。一个大规模的 CDN 拥有数万个服务器，分布于全球所有主要区域的几千个网络（ISP）。边缘系统从源系统下载内容（见图 10.1 的箭头 1），将相关内容缓存起来，并最终分发至客户端。在更复杂的边缘系统中，还会执行一些应用，在网络边缘处动态地生成内容，再将其发往客户端。

2. 监控系统

该系统实时监控互联网各处的“天气”情况和 CDN 所有部分（包括边缘服务器）的“健康”状况。在图 10.1 中，来自互联网的输入⑤中，既可能包括缓慢变化的信息（如来自成千上万个网络的 BGP 反馈），也有可能是快速变化的性能信息（使用 traceroute 和 ping 命令在互联网的成千上万个节点之间采集）。输入⑧包含边缘服务器、路由器及其他系统组成部分的详细信息，这些信息包括它们的活跃度、负载和资源使用状况。

3. 映射系统

映射系统的职能是将客户端定向到“最优”的边缘服务器，以下载所请求的内容（箭头⑥）。映射采用的常见机制是域名解析系统（DNS，箭头⑦）。在典型情况

下，根据一个内容提供商的域名 www. cp. com，需要在 CDN 的域上生成别名（即 CNAME' d），如 www. cp. com. cdn. net。CDN 托管域的一个客户端的域名服务器可以查询目标服务器的名称，据此得到其 IP 地址[10]。映射必须保证将每个客户端的请求转发到一个最优的目标服务器，该目标服务器具有如下特性：①目标服务器处于活跃状态，可能拥有被请求的内容且能提供内容服务；②目标服务器没有过载，其中考察负载的指标主要包括 CPU、内存、磁盘和网络的利用情况；③目标服务器和客户端之间的连接通畅，即没有或几乎没有丢包现象，往返延时也较小。为了做出正确的决策，映射系统把来自监控系统的网络动态和边缘服务器的状态（输入④）以及对互联网中各个域名服务器的流量估计作为输入信息，通过复杂的优化计算得出合理的映射方案。

4. 传输系统

该系统负责在互联网上远程传输数据。被传输内容的类型存在差异，对服务质量的要求也各不相同，这就使传输系统的设计极具挑战性。例如，与传输动态网页内容相比，从源（即编码器）服务器传输实时流媒体内容到边缘服务器就会有各种不同的需求。传输系统面临的挑战是设计一套小巧、易维护的通用机制和抽象模型，以满足多样化的需求。

5. 源系统

该系统是内容的源头，将内容提供给全球的客户端使用。一个庞大的 CDN 可能有几万个源系统（每个内容提供商有一个或多个），它们和 CDN 的其余部分发生交互。Web 源系统可以包括应用程序、数据库和 Web 服务器，提供流媒体服务的设施可以是存储点播内容（即预先制作的内容）并带容错机制的大型服务器，也可以是对实时内容进行视频采集、编码的设备。这些设施通常（但不完全是）由内容提供商管理，一般不包括数据中心。源系统还包括由 CDN 管理的入口，它是内容提供商的“命令中心”，用于提供和控制内容提供商的内容（箭头②和箭头③）。

10.1.2　传输系统

本节回顾几种不同类型的传输系统以及提高性能的优化方法，不同类型的传输系统可以根据被传输内容的端对端需求来区分。下面回顾一下传输系统采用的一些优化方法。

1. 实时流

实时流传输系统将实时媒体内容从流的源端（编码器）发送到终端用户，其目标是优化终端用户对于流数据的体验（见图 10.2）。编码器对实时流进行编码，并在实时流的持续过程中，将编码后的数据包按正确的顺序发送出去。这些数据流首先从编码器发到一个称为入口点的服务器集群。非常重要的一点是，编码器到入口点之间的网络延时要很低甚至没有（或几乎没有）丢包现象。编码器和入口点的连接一直处于监控中，如果连接出现问题或入口点由于某种原因失效，传输系统都会自动地将流转向另外一个运行良好的入口点。流数据从入口点再被发送给一到多个称为反射器的服务器集群。每个反射器依次将数据流发送到一个或多个边缘服务器集群。最后，每个终端用户通过映射系统从其附近的一个边缘服务器获取实时流数据。

传输系统的目标，是在发送实时流的过程中保证流的质量并最小化流数据的失

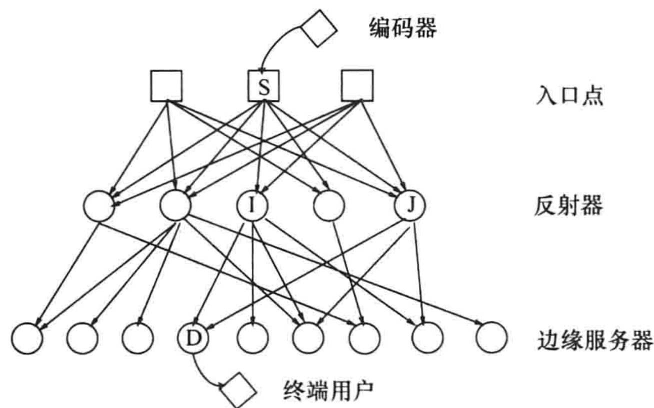


图 10.2　一个传送实时流的传输系统

真。终端用户感受到的数据失真包括流启动之前的长延时、信息丢失引起的音频/视频失真以及回放时的停顿现象。通过入口点和边缘服务器之间的反射器，每个流都自适应地选择一条或多条路径完成传送。例如，从入口点 S 出发的流在通过反射器 I 所在路径传输时，它的一个完全一样的副本也会同时沿着反射器 J 所在的路径传送，最终到达边缘服务器 D（见图 10.2）。如果一个数据包在一条路径上丢失，而它的副本通过另一条路径能够到达边缘服务器，那么该数据包仍可在边缘服务器得到恢复。而一种更复杂的技术是，利用一个编码方案将数据包编码，然后将编码后的数据流发送到多个路径，这样即使一些数据包在传输过程中丢失，在边缘服务器的解码过程中这些丢失的数据包仍然可以恢复。

另一种优化的例子是预迸发（pre - bursting），即流的初始部分在发送给终端用户时，其传输速率高于编码码率，这就可以使流数据快速填充终端用户媒体播放器的缓冲区，媒体播放器得以快速启动流，并减少回放过程中发生停顿的可能性。更多有关流传输系统算法和架构设计的问题可参见参考文献［6］和［12］。

2. Web 与在线应用

对于传输系统来说，它实现的是在源端和边缘端之间传送动态生成的内容。这种内容包括终端用户下载的动态网页，以及上传到网站的由用户生成的内容。这种传输系统的一个目标，是优化终端用户执行 Web 事务时的响应速度。与流媒体一样，传输系统需要用到一个或更多个中间节点，以便有效地将信息从源端传送到边缘端。传输系统也执行一些与应用有关的优化。例如，一个传输系统为了加速动态网页内容的传送，会从源端预先取出网页中的内嵌内容传送到边缘端，以降低源端和边缘端之间的通信延时。

与 IP 应用有关的一个传输系统，会着力于加速特定（非 http）的应用技术，如虚拟专用网（VPN）和 IP 电话。这类传输系统的架构设计，与为 Web 服务的传输系统相比在性质上有很大不同，区别主要在于终端用户体验所具有的高交互和实时性特点。

3. 覆盖路由方案

传输系统使用了一些与特定应用有关的增强方法来应对端到端需求。例如，作为一个节点，传输系统使用编码方式对丢包进行恢复，使用预迸发来加速流的启动，使用预取来加速下载[6, 12]，这些增强方法在提升传输系统整体性能中发挥了重要的作用。所有传输系统的一个基本功能，是需要在内容的发送点（源端或编码器等）与向终端用户提供服务的那个点（边缘端）之间寻找出一条“较好的路径”。这种纯网络层面的功能，是通过一种覆盖路由方案实现的，这种方案的实现是传输系统的一部分功能。

一个通用的覆盖路由方案是在每个起点 S（通常是源端）和每个终点 D（通常是边缘服务器）之间，计算一条或多条具有高可用性和低延时的“覆盖路径”。覆盖路由方案一般使用互联网的测量数据，对几百万个源—目标对计算它们之间的覆盖路径。通常情况下，由 BGP 确定的从源 S 到目标 D 之间的互联网路径（也叫直接路径）并不是这两个节点之间的最优路径。这并不奇怪，因为选择路由的互联网协议基于的大多是策略而非性能。一条从 S 出发，途径中间节点 I（通常是 CDN 的另一个服务器集群），然后到达 D 的间接路径㊀很可能更快捷或（且）可用性更高！覆盖路由方案利用了这一现象，选择最优的覆盖路径（直接或间接路径）来转发内容，从而增强了端到端的可用性以及终端用户的性能体验。全球性的覆盖路由方案所带来的收益是本章关注的重点。

10.1.3 本章的贡献

本章对全球性覆盖路由方案的性能和可用性收益进行了实验评估。很多学者对使用覆盖路由提升互联网的性能和可用性进行了研究[4, 11, 22]。不过，这些研究工作都或多或少地存在着以下局限：

1）以往的研究大多基于 Internet2㊁的平台，该平台的容量、使用模式、策略和目标都与商用互联网存在着很大的不同。

2）以前的研究中用到的覆盖主要是针对北美。然而众所周知的是，亚洲和欧洲在网络的互联性和关系上与美国大陆有很大不同。

本章给出了路由覆盖在商用互联网中对性能和可用性提升的第一个实验研究结果。本章使用 Akamai CDN 的一个全球子集来收集数据。具体地说，本章从 1100 个地域收集测量数据，这些地域分布于众多不同种类的 ISP，来自 6 个大洲，77 个国家，630 个城市。本章解决的问题是在位于互联网核心地带的源服务器和位于终端用户附近的边缘服务器之间寻找到一条最佳覆盖路径，其中既考虑了以往返延时为度量的性能，也考虑了路径的可用性，而那些以吞吐量为性能指标的应用（如大尺寸文件的下载）则不在本章的研究范围之内。

㊀ 必要时，一个间接路径的中间节点可能多于一个。原书注

㊁ Internet2 是一个高级组网协会，由美国几家主要的研究和教育机构组成。Internet②管理的 IP 网络可以用于研究目的。——原书注

本章的主要贡献为：

1）第一次采用商用互联网数据对覆盖路由进行评估。本章对一些当前使用的测试平台（如 PlanetLab[18] 和 RON[22]）提供了有价值的交叉验证，并指出：虽然这些平台为北美的商用互联网提供了性质上相似的数据，但它们并没有反映出网络拓扑（特别是在亚洲）的全球差异性。

2）分析结果显示，通过随机选择较少数量的冗余路径（欧美 3 条，亚洲 5 条），可以使可用性接近最优。另外，证明了在合理的采样间隔（如 10min）和冗余（2 条路径）下，除亚洲外有超过 90% 的源—目标对在延时指标上获得了改善，与理想值的差距缩小到 10% 以内。而对于源节点或目标节点在亚洲的路径，则需要 3 条冗余路径才能达到相同的性能。

3）本章提供的充足证据表明，覆盖路径的选择可以在很长的时段内（几个小时）保持固定不变，这说明对于几乎所有的源—目标对，网络信息测量的频度即使相对不太频繁也可以提供最好的性能。

10.1.4 内容安排

本章的余下部分组织如下：10.2 节对相关工作进行回顾，并介绍本章的研究背景；10.3 节描述本章的测试平台及如何收集测量数据；在 10.4 节和 10.5 节中，给出了在全球背景下通过路径覆盖可以获得的理想性能和可用性提升的详细度量；10.6 节解决覆盖的实用设计问题，探讨了覆盖在实用中的性能和可用性的结构、时域特性；在 10.7 和 10.8 节中给出了未来的研究方向及展望。

10.2 相关工作

在互联网的性能和可用性的度量方面，目前已有很多机构和项目进行了研究，如 CAIDA（Cooperative Association for Internet Data Analysis）[7] 和 NIMI（National Internet Measurement Infrastructure）[16, 17]。来自学术机构的路由覆盖网包括 MIT 的 RON（Resilient Overlay Network）项目[22]，以及华盛顿大学的 Detour 项目[11]。Akamai 公司提供的商用分发服务将覆盖路由应用于实时流、动态网络内容和应用加速[1]。

Andersen 等人[5]给出了一个称为 RON 覆盖路由的实现和性能分析。他们发现，其方法可以使延时性能提升 51%，与本章在北美环境下获得的 63% 的改善效果相当。Akella 等人[2]将一个更简单的路由控制多路寻址方案与覆盖路由方案进行了比较。虽然其研究的着眼点与本文不同，但它提供了一个单路寻址站点在默认情况下的结果，发现如果用往返延时来衡量，覆盖路由可以使性能平均提高 25%。他们的实验基于美国 17 个城市的 68 个节点，这个实验环境与本章中的北美 110 个节点相当，而本章中的方法可以将整体延时性能提升约 21%。然而，性能的提升在其他大陆之间有显著不同。Savage 等人[23]使用了 20 ~ 40 个节点的数据，发现大约 10% 的源—目标对的最佳覆盖路径的延时较直接路径低 50%。而在本章的北美节点实验中，虽然对于其他大陆的节点对而言结果也会有很大的不同，但就北美节点来说这个源—目标对的比

例为 9%，在数值上也与他们的结果基本相当。与本章的评估相似，Gummadi 等人[13]在 PlanetLab 平台上实现了随机单跳源路由，其结果表明使用至多 4 个随机选择的中间节点，就可以提升互联网路径的可靠性。

10.3　实验环境的配置

本节介绍采集数据时的实验环境配置，可用于优化边缘网络（终端用户处）和企业源服务器之间的互联网路径。终端用户通常位于小型、较低层的网络中，而企业源服务器通常位于一级网络。本节在研究路由覆盖问题时，涉及的节点只包括部署在大型、一级网络中的节点，这些节点在从源（企业源服务器）到目标（边缘服务器）的间接路径中，起到了中转节点的作用。

10.3.1　测量平台

Akamai 的 CDN 服务器部署于几千个地域和网络的服务器集群中。这些集群大部分位于互联网边缘附近，即靠近终端用户的非一级 ISP 网络中。小部分集群部署在直接位于一级提供商处的核心 ISP 附近，即那些适合企业源服务器部署的位置。基于地理和网络位置的多样性以及安全性等其他考虑，本节实验从整个 CDN 中选择了 1100 个集群，这些集群分布于 6 个大洲 77 个国家的 630 座城市。每个集群中的服务器与单个内容提供商连接。大约 15% 的集群位于核心地带，其余则分布在网络边缘，而间接路径中的中间节点位于接近核心的集群。表 10.1 显示了所选节点的地理分布。实验中，所有的数据采集活动都与 CDN 正常的数据收集活动完全隔离。

表 10.1　实验平台中节点的地理分布

大洲（助记符）	边缘集	核心集
非洲（AF）	6	0
亚洲（AS）	124	11
中美洲（CA）	13	0
欧洲（EU）	154	30
北美洲（NA）	624	110
大洋洲（OC）	33	0
南美洲（SA）	38	0

10.3.2　性能和可用性数据的收集

在 1100 个集群中的每个集群上运行一个任务，该任务每隔 2min 向核心集中的每个节点发送一个大小为 64B 的 ICMP 回显请求（ping），这样一个核心节点处的请求速率保持在小于 10 个/s，每个任务持续 1.5h。如果发生丢包，即 10s 内无响应，则上报一个特殊值作为往返延时。所有的集群每天运行三次该任务，运行时间选在东亚、欧洲和北美东海岸的网络流量高峰时。实验从 2004 年 10 月 18 日开始，一共运

行了4周。这样，在实验中每条路径被探测了3780次，而总探测次数大约为65200万次。由于网络维护或设施变动等原因，核心集的一小部分节点在某个时段不可用，这时需要对数据进行过滤并清除涉及这些节点的所有数据。使用标准全节点对(all - pairs) 最短路径算法[9]的一个修正版本对数据集进行分析，以确定包含1个、2个及3个中间节点（来自核心集）的最短路径。经过处理，可以得到一个7元组的集合，该7元组的形式为 <时间戳，起点号，终点号，直接往返时间，单跳最短往返时间，2 - 跳最短往返时间，3 - 跳最短往返时间>。根据起点和终点所在的大洲，这个7元组可分成若干类。

如果连续3个或更多个ping操作没有回应，则认为一条路径已经不可用。Akella等人[2]使用了相同的配置，而ping命令发送的间隔为1min。假设丢包之间相互独立，且2min间隔内丢包概率的数量级为1%，那么由于随机拥堵造成的连续三个ping命令丢失的概率为10^{-6}的数量级。路径不可用的时段从发送第一个被丢失的ping命令开始算起，至最后一个连续丢失的ping命令的发送时间为止。这看上去是个保守的估计，意味着所做结论只能针对超过6min的互联网路径失效。

在可用性分析时过滤掉了一些区域内边缘节点的测量数据，因为其测量数据的失效特性与其他的互联网路径相比具有非常明显的差异。

10.3.3 评估

本章对结果进行了汇总，汇总工作基于源节点和目标节点所在的大洲，这么做的原因是，企业网站的经营策略一般是针对来自他们所在大陆的用户。这些洲际路径的分类可以用一个助记符来表示，如AS - NA（见表10.1）表示边缘服务器位于亚洲、源服务器在北美。

10.4 覆盖路由对性能的改善

本节评估覆盖路由在理想情况下的性能提升。对于每个源—目标对，两者之间所有可能的间接路径都予以考虑，并实时计算出最佳间接路径。本节使用延时作为性能的测量标准，即从源节点到目标节点的往返时间（简写为RTT)。

对于每个源—目标对，本节比较了直接路径和最快间接路径，结果在表10.2中给出。根据路径类别和最快间接路径较直接路径的延时所提升的百分比，将数据集分成几组。表10.2显示了每组数据中源—目标对的所占比例，表中的每行之和为100%。例如，对于表中AS - AS那一行，“ < -10%”那一组表示最快间接路径比直接路径慢至少10%，且所有AS - AS路径中属于该组的占15.5%。“±10%”那组表示，最快间接路径和直接路径性能相近，二者之间的延时差异在10%以内，且所有AS - AS路径中属于该组的占24.7%。在剩下的直接路径中，有23.4%获得了较小提升（10% ~30%)，有13.2%提升明显（30% ~50%)，另有23.2%的路径获得了很大的改善，采用间接路径可以使延时性能提升一半以上。

表 10.2　延时下降的比例

分类	< -10%（较小）	±10%（相当）	10% ~30%（较大）	30% ~50%（显著）	>50%（很大）
AF - AS	4.0	44.5	44.2	5.7	1.6
AF - EU	0.6	69.3	18.1	9.7	2.3
AF - NA	0.0	74.2	21.6	3.5	0.6
AS - AS	15.5	24.7	23.4	13.2	23.2
AS - EU	0.9	33.9	45.5	12.5	7.2
AS - NA	0.1	43.2	42.4	7.6	6.7
CA - AS	0.0	40.5	53.5	4.6	1.4
CA - EU	1.4	53.2	42.3	2.5	0.7
CA - NA	1.7	44.1	41.3	11.2	1.8
EU - AS	0.6	24.5	63.8	7.8	3.2
EU - EU	10.5	36.4	30.5	12.6	10.0
EU - NA	0.0	50.6	45.1	3.3	0.9
NA - AS	0.0	34.0	57.9	5.4	2.6
NA - EU	0.1	43.1	51.1	4.4	1.4
NA - NA	2.4	34.7	39.0	15.0	9.0
OC - AS	6.1	38.9	18.9	22.9	13.2
OC - EU	0.0	60.4	35.1	3.9	0.7
OC - NA	0.0	66.7	25.6	6.3	1.4
SA - AS	0.1	43.1	47.9	5.5	3.4
SA - EU	0.4	66.1	28.9	2.3	2.2
SA - NA	0.9	55.1	35.1	5.7	3.3

从整体上看，按分类计算有大约 4% ~35% 的源—目标对的延时指标提升了 30% 以上。另外，AS - AS 和 EU - EU 两类中大量源—目标对的性能提升超过了 50%，这表明在这些洲的 ISP 之间存在着大量的病态路由。AS - AS 路径中有很多都经过加利福尼亚的对等位置，如台北的超媒体和上海的中国电信之间的路径。采集到的路由信息显示，所有从台北超媒体至亚洲其他地方的路由都途经加利福尼亚，一个例外是台北超媒体至上海的中国电信，它是直接从台北连到了上海。因此，台北至上海的路径几乎得不到提升，因为相比直连而言所有的其他替代路径当然都是绕远的。与此同时，台北的超媒体至亚洲其余地域的路径则获得了很大的性能提升，因为它们之间的直接路径本身就非常绕远，不过其中倒是存在一条途径上海中国电信的路径，它的传输速率快 50% 以上。

企业很希望提高其网站在最坏情况下的性能，以提高用户的访问速度。所以，通

过覆盖路由对上述各类中的最长路径延时进行改善就特别具有吸引力。从一个给定的类别中选择一个“典型的”源—目标对和一个连接性较差的源—目标对，将两者的延时降低情况进行了比较。按照直接路径的延时，将每类数据按 10ms 的间隔分组，然后提取典型的源—目标对的第 50 百分位数分组和连接较差的源—目标对的第 90 百分位数分组。使用这些分组的数据计算最快间接路径较直接路径的平均提升。表 10. 3 比较了每类中典型的源—目标对和连接较差的源—目标对之间的性能提升。对于典型的源—目标对，在表 10. 2 所示的 21 个类别中，只有 AS - AS、OC - AS 和 CA - NA 在延时方面的提升超过了 20%。相比而言，对于连接较差的源—目标对，有一半的类别得到了超过 20% 的性能提升，其中 AS - AS、AS - NA 和EU - EU等重要的类有显著的改善，而与此形成对照的是，对于源自非洲的源—目标对，第 90 百分位数分组的延时不仅很高，覆盖路由对其也没有产生作用。对于AS - AS，典型的和连接较差的源—目标对都有显著改善，而前者的提升更为明显。不管怎样，大体上可以得到这一结论：与典型的源—目标对相比，连接较差的源—目标对可以得到更显著的改善。

下面进行更深入的评估，分析连接较差的源—目标对中较大提升、显著提升以及很大提升各占的比例。从一个给定类别中，选出直接路径延时超过该组第 90 百分位数的那些源—目标对，将其分组并生成连接较差的源—目标对的延时降低量的柱状图，如图 10. 3 所示。在该图中，同时显示了该类中所有源—目标对的对应数值（注意：在该图中，所有源—目标对的数据在表 10. 2 的最后三列给出）。在连接较差的源—目标对中，超过 80% 获得了较大及较大以上的性能提升，而 67% 的样本则有显著或巨大的性能提升。有些类别与上述结果存在着偏差，例如目标在非洲的连接很差的源—目标对并未获得大的性能提升。

表 10. 3　典型的和连接较差的源—目标对的延时降低情况

分类	第 50 百分位数			第 90 百分位数		
	直接 /ms	最快 /ms	降低 (%)	直接 /ms	最快 /ms	降低 (%)
AF - AS	350	290	17	740	700	5
AF - EU	150	120	20	620	620	0
AF - NA	200	180	10	560	550	2
AS - AS	230	110	52	590	350	41
AS - EU	320	260	19	500	360	28
AS - NA	230	200	13	470	280	40
CA - AS	230	200	13	300	250	17
CA - EU	160	140	12	200	170	15
CA - NA	90	70	22	130	100	23

（续）

分类	第 50 百分位数			第 90 百分位数		
	直接 /ms	最快 /ms	降低 (%)	直接 /ms	最快 /ms	降低 (%)
EU－AS	300	260	13	390	300	23
EU－EU	30	30	0	80	60	25
EU－NA	130	120	8	190	160	16
NA－AS	190	160	16	260	210	19
NA－EU	130	110	15	180	150	17
NA－NA	50	40	20	90	70	22
OC－AS	200	140	30	340	220	35
OC－EU	330	300	9	400	330	17
OC－NA	220	200	9	280	230	18
SA－AS	320	280	12	470	340	28
SA－EU	230	210	9	290	250	14
SA－NA	160	150	6	240	190	21

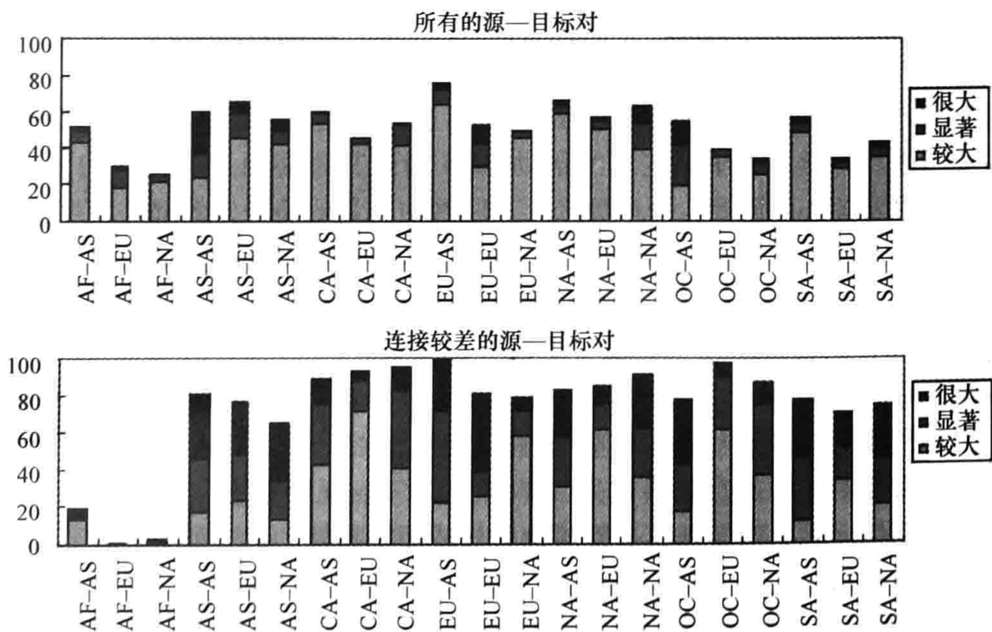


图 10.3　所有的和连接较差的源—目标对的延时下降情况

10.5　覆盖路由对可用性的提高

在本节中，对理想情况下覆盖路由带来的可用性改善进行评估。在评估过程中，

每个源—目标对之间所有可能的间接路径都将予以考虑，只要一条间接路径可用则实时选用它以缓解失效。

本节研究了每个源—目标对之间直接路径的失效频度，以及在直接路径失效期间每条间接路径发挥替代作用的时间占比，这有助于分析覆盖路由对可用性的最大改善效果。图 10.4 显示了每个类别中源—目标对之间的直接路径出现失效的比例。从图中可以发现，直接路径的失效比例从 0.03% 到 0.83% 不等，其中亚洲的可用性最低：在失效率最高的 10 个类别中，有 9 个存在着位于亚洲的端点。而使用覆盖路由后，大多数类别的失效比例下降了 0.3% ~0.5%，这说明间接路径有助于避免直接路径的失效。事实证明，那些与亚洲有关的高失效率类别的可用性得到了显著提升。

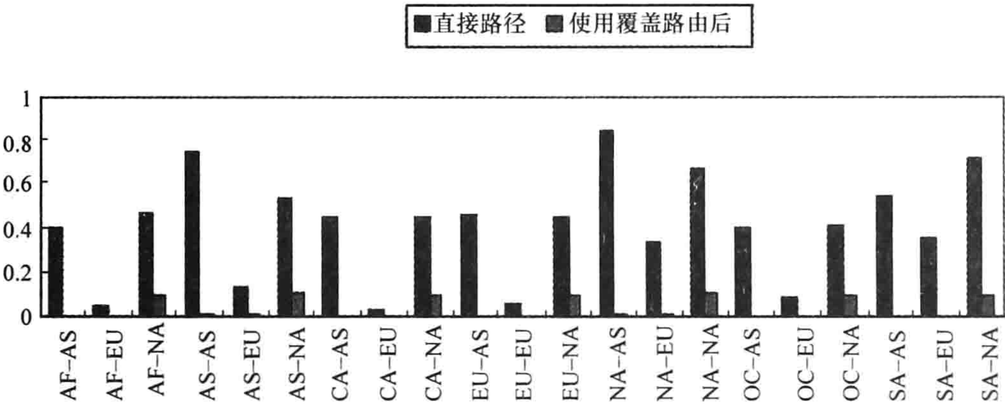


图 10.4 使用覆盖路由后失效比例的下降

与 10.4.1 节类似，本节研究覆盖路由如何提高那些直接路径经常失效的源—目标对的可用性。同样，对于那些有意借助 CDN 增强其终端用户可用性的公司来说，这也是它们非常关心的问题。众所周知，互联网中大量的路径失效都是发生在一小部分路径上。参考文献［15］的研究结果显示，互联网中 30% 的失效发生在 3% 的路径。在表 10.4 所示数据中也可以发现类似的情况：大约 30% 的失效发生在 3% 的直接路径，而 10% 的直接路径"贡献"了 50% 的失效。

数据显示，每类中可用性最低的那些源—目标对产生的失效数占总失效数的 30%，因此本节对这些源—目标对重新进行了可用性分析，结果如表 10.4 所示。如果一个源—目标对之间的直接路径失效率超过 20%，证明在这条直接路径上存在某些特定的长期性问题，而非随机、瞬时的失效或短期拥堵。几乎所有存在长期可用性问题的源—目标对在使用了覆盖路由后，都获得了很满意的可用性！改善那些可用性最低的源—目标对，是覆盖路由的一个关键作用。

表 10.4 连接性较差路径的可用性数据

类别	30% 失效率的路径百分比	无覆盖路由的失效率百分比	有覆盖路由的失效率百分比
AF - AS	4.5	25.8	0
AF - EU	1.7	8.8	0
AF - NA	0.6	36.2	0

（续）

类别	30% 失效率的路径百分比	无覆盖路由的失效率百分比	有覆盖路由的失效率百分比
AS – AS	2. 7	31. 4	0
AS – EU	1. 5	9. 8	0
AS – NA	0. 4	30	0
CA – AS	3. 5	28. 2	0
CA – EU	1. 6	10. 9	0
CA – NA	0. 5	30. 3	0
EU – AS	3	30. 1	0
EU – EU	0. 9	10. 8	0
EU – NA	0. 4	30. 1	0
NA – AS	2. 7	32. 3	0
NA – EU	0. 4	13. 2	0
NA – NA	0. 2	40. 2	0
OC – AS	3. 1	30. 8	0
OC – EU	1. 4	10. 7	0
OC – NA	0. 4	29. 3	0
SA – AS	3. 3	28. 8	0
SA – EU	2. 2	9. 5	0
SA – NA	0. 8	23	0

10. 6　在实际设计中获得提升

10. 4 节和 10. 5 节中的分析描述了理想情况下的特征，即使用网络测量数据实时计算节点之间的间接路径，且假设被探测出来并用做间接路由的间接路径的数量不受限制。因此，上述对覆盖路由提升性能和可用性的分析是一个基于最好情形下的估计。然而，在实际系统中，在给定时刻 t 得到的测量数据需要在 $t+\tau$ 时刻之前完成覆盖路径的计算，以供传输系统使用。而对于一个给定的源—目标对，在任意给定时间内构造出并供使用的间接路径的数量（路径数目 κ）并不多。本节将这些现实因素引入到分析中，并评估其对结果的影响。当 κ 增大且 τ 减小时，覆盖路径的构建成本会上升，但同时会要求其质量也有所提高，并接近 10. 4 节和 10. 5 节中给出的最佳路由。

首先，评估一个简单的多路径、无记忆覆盖路由方案，该方案使用单纯的静态信息随机选择 κ 条路径中的部分路径，并使用这些路径对内容进行路由。这种覆盖路由方案当然有可能比理想方案要差一些，但本节的目标是提供一个标本来验证在使用覆

盖路径时使用智能和自适应技术的重要性。令人惊讶的是，结果显示这种随机选择很成功，当 $\kappa=3$ 时获得了接近最优的可用性。这表明，互联网提供了非常好的路径多样性，一般来说，失效率也较低。然而，上述策略在性能的提升方面却显得无能为力。这表明，若想提高性能，在构造覆盖路由时就必须要仔细选择路径，这就是本节随后部分的关注焦点。

10.6.1 最优路径的稳定性

对于一个寻求性能最优的覆盖路由方案，在选择路径时往往会偏离最优选择。这是因为，各条路径上的延时会随时间的推移而发生变化，从而使最优路径发生了变化。为便于分析，首先需要了解源—目标对的特点。所有的源—目标对一般可以归于下面两类：

1）从源节点到目标节点（以下分别简称源和目标）的最好路径相当持久，无论两者之间所有路径的延时如何变化，最优路径都不会改变。

2）随着时间的推移，路径的延时变化将导致源和目标之间的最好路径发生明显的变化，反过来又会引起最佳路径的变化。

第一类中的源—目标对不需要非常动态的覆盖设计来选择间接路径。例如，对于由新加坡 Pacific Internet 至伦敦 AboveNet 的那条路径，其直接路径从新加坡经东京、旧金山、达拉斯和华盛顿特区到达伦敦，花费时间大约 340ms。而两者之间有一条途经某个中间节点的间接路径，该中间节点位于伦敦的 ISP Energis Communications。新加坡 Pacific Internet 与该中间节点之间只有一跳（可能是卫星链路），其延时为 196ms，而随后从该中间节点到伦敦 AboveNet 只需 2ms。因此间接路径比直接路径快了 140ms 以上（即 41.2%）。即使延时发生了变化，但最优路径几乎不会变化。

对于第二类的源—目标对，延时的差异则非常重要。通过计算一个统计量（命名为 Churn），本节系统地检验了路径间延时的变异程度。该统计量可以测量最好的 κ 条路径的集合在两个不同时刻的差异程度。对于给定的节点对，在每个时刻 t 有

$$Churn_t(\kappa,\tau) \overset{\Delta}{=} |S(\kappa,t) - S(\kappa,t+\tau)|\kappa$$

式中，$S(\kappa, t)$是在 t 时刻源—目标对之间所有节点中性能最好的 κ 条路径的集合。对于一个节点对来说，统计量 $Churn(\kappa, \tau)$是对 t 的所有有效值计算出的 $Churn_t(\kappa, \tau)$的平均值。$Churn(\kappa, \tau)$的值介于 0 和 1 之间，为 0 时意味着具有一组持续不变的最好路径；接近 1 时则表示这些最好路径存在着快速的变化。研究发现，大多数源—目标对的 $Churn(\kappa, \tau)$都超过 10%，即使选择了多达 $\kappa=5$ 条最好路径及预测时间 $\tau=2$min 时也是如此。

为了更细致地检验路径扰动，可以定义一个松弛度量 $RelaxChurn(\kappa, \tau)$，该量仅考虑路径 $\pi \in S(\kappa, t) - S(\kappa, t+\tau)$，这些路径在 $t+\tau$ 时刻的延时高于 $S(\kappa, t+\tau)$中延时指标最差的路径的延时的 110%。也就是说，如果保留 π 中的路径则 $t+\tau$ 时刻的性能将降低 10% 以上。有意思的是，对于大多数类别中 80% 以上的源—目标对，$RelaxChurn(\kappa, \tau)$平均而言都小于 10%。这表明，如果路径选择算法能够基于当前的测

量值对未来做出预测，则可以接近理想的性能。

图 10.5 中显示了在 $\kappa=1$ 和 $\tau=2\text{min}$ 时，*Churn*(κ, τ)和 *RelaxChurn*（κ, τ）小于 10% 的源—目标对的百分比。从图中可以发现，当路径的两个端点都在亚洲时，路径的 *RelaxChurn* 的值比 *Churn* 的值更高，但只有 63% 的源—目标对存在 *churn* 值较低的路径。所以，对于 AS – AS 路径，只有在付出更高的网络测量成本时，性能才有可能提升。

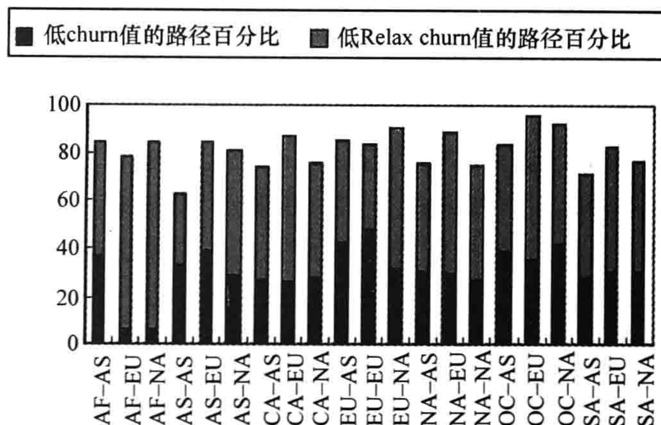


图 10.5　$\tau=2\text{min}$ 和 $\kappa=1$ 时具有低 *Relaxchurn* 值和低 *Churn* 值的源—目标对的百分比

10.6.2　预测覆盖路由的性能提升

在 10.6.1 节的分析中，是利用纯结构特性来检验稳定性。在本节，将使用参数 κ 和 τ 的覆盖路由与一直选择最优路径的理想情况进行性能的比较。注意，因为给定时间的最优路径至少与直接路径一样快，所以这将使覆盖处于一个较高的标准。

表 10.5　与最优延时的差异在 10% 以内的路径百分比

类别	路径所占百分比			
	$\kappa=1$ $\tau=2$	$\kappa=1$ $\tau=10$	$\kappa=2$ $\tau=2$	$\kappa=3$ $\tau=2$
AS – AS	62.4	59.5	84.6	89.4
AS – EU	76.2	74.1	92.2	94.5
AS – NA	74.8	71.6	94.0	96.0
EU – AS	74.4	72.3	88.4	92.8
EU – EU	80.1	78.1	91.6	93.1
EU – NA	83.0	82.2	94.7	96.2
NA – AS	68.1	66.2	88.8	93.7
NA – EU	82.3	81.3	95.4	97.2
NA – NA	71.6	69.6	92.0	95.0

一个需要验证的常见情形是当 $\kappa=1$ 的情况，也就是说在以后的评估中只选择最优路径。表 10.5 中的第 2 列和第 3 列显示的是当 τ 分别为 2min 和 $\tau=10\text{min}$ 时的结果。以类别 NA – NA 为例，从表中可以看出，当 $\tau=2\text{min}$ 时，71.6% 的路径的延时

与最优延时相差10%以内。甚至当τ长达10min时，仍有69.6%的路径取得了相同的结果。与源自欧洲的路径相比，源自亚洲的路径又一次大大偏离了最优值，而源自北美的路径数据差异范围最大。

既然当$\kappa=1$时性能的提升在各地都不明显，那么就赋予κ更高的值。以NA-EU为例，如表10.5所示，当$\kappa=1$时有82.3%的路径和最优值相差10%以内；将κ增加到2后，相差10%以内的路径数增加了大约13.1%；κ增加到3时，其余的路径只获得了很小的提升，相差10%以内的路径数只进一步增加了1.8%。$\kappa=2$相对于$\kappa=1$获得的边际收益并不成比例，而$\kappa=3$时获得的边际收益则低得多。事实上，除了目标点在亚洲的路径，当$\kappa=2$时超过90%的源—目标对与理想性能的偏差在10%以内，甚至τ在增加时该事实依然存在。结果也表明，在使用1或2条路径时($\kappa=1$、2)，覆盖路由可以取得满意的效果。例如，对于NA-EU中的源—目标对，当$\kappa=2$时有95.4%的节点对与最优值相差10%以内。通过与前面$\kappa=1$时的结果相比可知，在同样的条件下既然有82.3%（$\kappa=1$）的节点对只需要选择1条路径，那么可想而知一个覆盖路由方案使用两条路径时可能只是为了满足额外的那13.1%的节点对，平均而言，每个节点对只使用1.09条路径。

两个端点都在亚洲的源—目标对表现出了不同的特点。例如，当κ从1增到2时，与最优值偏差10%以内的源—目标对的比例从62.44%跃升到84.57%（这时，加权平均后每个节点对使用的路径数为1.31）。然而，这一比例要想达到接近90%的话，需要κ增加到3。

表10.5显示的是$\tau=2$min和$\kappa=2$时的结果。需要注意的是，当τ在2~10min之间变化时，这些数值仍然保持相对稳定（与$\kappa=1$时类似）。这意味着，在路径数量很大时，提高测量频度不会带来延时性能的提高。在10.6.3节中将进一步研究延时性能对τ的敏感性。

相对理想情况而言，旨在提高覆盖路由性能的设计反而降低了可用性。这是因为，如本章前面几个例子所示，所有路径中效果较好的那些路径一般会共享一小部分共用链路，这不仅会导致路径多样性变小，而且当这些共享的链路同时失效时会暴露出更大的脆弱性。

10.6.3 稳定性

10.6.2节的分析表明，当τ的取值在2~10min之间时，覆盖路由带来的性能提升对τ的取值具有轻微的敏感性。在本节，采用一些极端的例子研究基于预测的覆盖路由的时间敏感性。在每天内每隔一定时间（4~11h不等）采样一个1.5h的样本，在该样本中使用基于测量的覆盖路由，再评估下一个1.5h样本中的性能。尽管在两个样本之间的时段内覆盖路由完全有可能是次优的，但结果显示即使对这么长时间跨度进行预测，NA-NA中大约87%和AS-AS中大约74%的路径与理想情况下的偏差仍在10%以内。数据结果显示，对于大多数的节点对而言，各种可选路径的性能具有高度的一致性。相反，只有少量路径在短期内变化较大，即便κ上升到5或6，使用预测机制的覆盖路由也很难对这些路径实现优化[20]。

10.7　未来研究方向

本章定量地研究了覆盖路由对性能和可用性的改善，以及这种改善在不同洲之间的差异。互联网低效率的根源在于每个 ISP 的行为都是本着其自身的经济利益考虑，这种状况仍会继续长期存在。更进一步地说，地域间的行为模式的巨大差异只是一个表象，其根源可能在于更深层的结构特性，而且也不一定能随着时间的推移因连接性和经济性的改善而消失，这就要求 CDN 覆盖网能够路由越来越多的流量。随着覆盖路由优化越来越多地被采纳，优化对提供底层服务的 ISP 的影响以及这些 ISP 在其网络中进行的优化都成为未来研究的热门话题[8, 14, 19]。

10.8　写给使用者

经过十年的发展，互联网在技术、媒体、娱乐、商务、软件以及政府等各个层面上都采用了 CDN 技术，CDN 无疑在促进互联网业务方面扮演了一个中心角色。年复一年，经 CDN 技术加速的流量继续呈现跳跃式增长。对网站、流媒体、应用的性能和可用性进行增强是一个双重挑战，这已成为过去十年来 CDN 技术演化的一个基本驱动力。在结束本章讨论之前，本节将目光重新投向这些挑战以及 CDN 未来的发展道路：

1）对于互联网上实现数十亿元销售额的商家，如果其网站在流量的高峰时刻停机 10min，那么带来的损失将数以百万元计，并可能给用户带来极坏的印象[24]。不仅如此，电子商务的收入正以显著的速率增长，预计每隔两三年就会翻一番。越来越多的证据表明，网页的快速下载可以促使电子商务网站的更大发展，并带来更高的收入。这就需要实现很少停机或没有停机的更高目标，以便将内容更好地分发到互联网上。

2）对于一些大的媒体和娱乐公司，它们通过互联网将内容分发给大量的终端用户。用户喜欢网络流媒体，是因为它可以提供给用户任何感兴趣的内容，同时用户也需要一种真实的、与电视一样的体验，如视频能快速启动、播放时从不出现停顿现象，这就需要以比传统方式更高的性能在互联网上分发内容。

3）作为主要的娱乐和新闻来源，随着互联网的地位越来越稳固，一些内容提供商正面临着瞬时拥塞的问题，这就需要互联网能以一种灵活的方式，在不降低可用性和性能的前提下将内容分发给终端用户。

4）业务外包、远程协作以及政府通信等新的业务趋势，要求制订更明确的性能和可用性标准，这些标准不只是局限于一个国家或一小部分国家，而应该是全球性的。通过互联网形成一个大的虚拟团队，其成员即使分布于全球也能实时地在一个项目中分工合作，这种情况正在变得越来越普遍。不仅如此，互联网中很多新应用具有比以往更严格的性能要求。交互类应用（如虚拟专用网络上的远程外壳和多用户游戏）和新兴技术（如 IP 电话）都对延时具有高度的敏感性。这就需要满足新的、更严格的可用性要求和性能要求，以支持下一代的互联网应用。

这些挑战将继续推动该领域向前发展，并对 CDN 的发展产生深远的影响。

致谢

本章中的实验结果来自参考文献［20］和［21］，作者 Ramesh Sitaraman 的部分研究工作受国家科学基金资助（CNS－0519894）。

参考文献

[1] Akamai Technologies, Inc. http://www.akamai.com.

[2] Akella, A., Pang, J., Maggs, B., Seshan, S., and Shaikh, A. A comparison of overlay routing and multihoming route control. In *Proceedings of the ACM SIGCOMM*, pp. 93–106, Portland, OR, August 2004.

[3] Akella, A., Maggs, B., Seshan, S., Shaikh, A., and Sitaraman, R. A Measurement-Based Analysis of Multihoming. *Proceedings of the 2003 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, August 2003.

[4] Andersen, D. G. *Improving End-to-End Availability Using Overlay Networks*. PhD thesis, MIT, 2005.

[5] Andersen, D. G., Balakrishnan, H., Kaashoek, F., and Morris, R. Resilient Overlay Networks. In *18th ACM SOSP*, Banff, Canada, October 2001.

[6] Andreev, K., Maggs, B., Meyerson, A., and Sitaraman, R. Designing Overlay Multicast Networks for Streaming. *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, June 2003.

[7] Home page of the Cooperative Association for Internet Data Analysis (CAIDA). http://www.caida.org.

[8] Clark, D., Lehr, B., Bauer, S., Faratin, P., Sami, R., and Wroclawski, J. The Growth of Internet Overlay Networks: Implications for Architecture, Industry Structure and Policy. In *33rd Research Conference on Communication, Information and Internet Policy*, Arlington, Virginia, September 2005.

[9] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. Introduction to Algorithms. MIT Press and McGraw-Hill, 2001.

[10] Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., and Weihl, B. Globally distributed content delivery. *IEEE Internet Computing,* September 2002, pp. 50–58.

[11] Home page of the Detour Project http://www.cs.washington.edu/research/networking/detour/.

[12] Kontothanassis, L., Sitaraman, R., Wein, J., Hong, D., Kleinberg, R., Mancuso, B., Shaw, D., and, Stodolsky, D. "A Transport Layer for Live Streaming in a Content Delivery Network". *Proceedings of the IEEE, Special issue on evolution of internet technologies,* pp. 1408–1419, Vol. 92, Issue 9, August 2004.

[13] Gummadi, K. P., Madhyastha, H., Gribble, S., Levy, H., and Wetherall, D. Improving the Reliability of Internet Paths with One-hop Source Routing. In *Symposium on Operating System Design and Implementation (OSDI)*, San Diego, CA, 2003.

[14] Keralapura, R., Taft, N., Chuah, C., and Iannaccone, G. Can ISPs take the heat from overlay networks? In *ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, 2004.

[15] Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C.-N., and Diot, C. Characterization of failures in an ip backbone. In *IEEE Infocom*, Hong Kong, 2004.

[16] Home page of the National Internet Measurement Infrastructure (NIMI). http://ncne.nlanr.net/nimi/.

[17] Paxson, V., Mahdavi, J., Adams, A., and Mathis, M. An Architecture for Large-Scale Internet Measurement. *IEEE Communications*, August 1998.

[18] Home page of PlanetLab. An open platform for developing, deploying, and accessing planetary-scale services, http://www.planet-lab.org/.

[19] Qiu, L., Yang, Y. R., Zhang, Y., and Shenker, S. On Selfish Routing in Internet-Like Environments. In *ACM SIGCOMM*, 2003.

[20] Rahul, H., Kasbekar, M., Sitaraman, R., and Berger, A. Towards Realizing the Performance and Availability Benefits of a Global Overlay Network. *MIT CSAIL TR 2005-070*, December 2005.
[21] Rahul, H., Kasbekar, M., Sitaraman, R., and Berger, A. Towards Realizing the Performance and Availability Benefits of a Global Overlay Network. *Passive and Active Measurement Conference*, Adelaide, Australia, March, 2006.
[22] Home page of the Resilient Overlay Networks (RON) project. http://nms.csail.mit.edu/ron/.
[23] Savage, S., Collins, A., Hoffman, E., Snell, J., and Anderson, T. The End-to-End Effects of Internet Path Selection. In *Proc. ACM SIGCOMM*, pp. 289–299, Cambridge, MA, 1999.
[24] Zona Research. The need for speed II. Zona Market Bulletin 5 (April 2001).

第3部分 先进CDN平台与应用

第11章 缓解瞬时拥塞的动态CDN

Norihiko Yoshida

11.1 引言

随着互联网信息量的快速增长和网络浏览者随时随地接入网络，瞬时拥塞（flash crowd）这个互联网上新的拥塞现象已成为传统技术难以解决的问题。瞬时拥塞指在特定Web站点突然出现的不可预计的请求流量瞬间暴涨现象。与持续的互联网拥塞不同的是，瞬时拥塞会带来短期拥塞现象，造成Web站点过度配置、托管CDN效率低下和资源浪费。因此，瞬时拥塞对互联网提出了新的挑战。

“瞬时拥塞”这个术语最早出现于科幻小说作家Larry Niven于1973年创作的短篇小说《瞬时拥塞》[31]。在小说中，价格低廉和方便易用的心灵传输方式使得全球数以万计的人在瞬间同时涌向任何感兴趣的场景地点，导致了无序和骚乱。

该术语随后于20世纪90年代后期被用于描述互联网上的类似情况。当一个Web站点受到大量用户的访问时，往往会导致出人意料的流量激增，使得网络迅速饱和及服务器瘫痪，从而导致该网站暂时无法访问，有时候也称为“SlashDot效应”[2]或“Web热点”[50]。一个瞬时拥塞的例子如图11.1所示[34]，该事件发生在2005年11月3日的“LIVE！ECLIPSE”网站[26]。

瞬时拥塞并不常见，它与一般的时变工作负载（如每日时段效应[13]）不同。例如，很多人喜欢在午餐时间浏览网页，而这种长期的周期性趋势可以被预测出来。然而，瞬时拥塞很容易发生，例如所有用户同时对某个热门网站快速浏览片刻就能产生一个瞬时拥塞。因为发生频率越来越高以及整体上的不可预见性，瞬时拥塞已经成为大多数网站的棘手问题。

一个传统CDN在请求负载相对稳定的情况下能够正常工作。然而，这种CDN被称为静态CDN，原因在于它一直在使用固定数量的代理缓存服务器（surrogate）工作，并且它一直处于待机状态。如果都按应对瞬时拥塞的要求使用高级别的资源配置，就会导致资源效率低下。因为高负载的罕见性，静态CDN会使资源的利用效率低下，并且大多数代理缓存服务器多数时间内都处于空闲状态。

另外，Web请求可以是瞬间爆发式的。预测瞬时拥塞的负载峰值非常困难。即

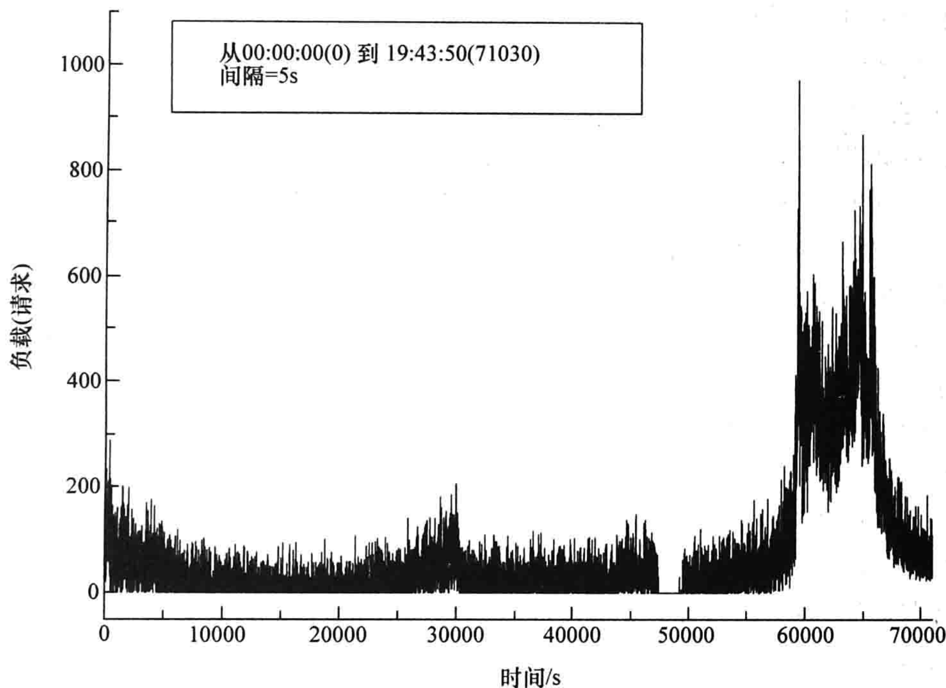


图 11.1　2005 年 11 月 3 日“LIVE！ECLIPSE”站点发生的瞬时拥塞

使是一个配置完善的 CDN 站点，都有可能因为不可预见的流量激增而瘫痪。

为了解决瞬时拥塞问题，目前已提出了若干解决方案。一个可行的方式应当遵循这样一个原则：将静态 CDN 改造为自适应性的动态 CDN。这样一来，网站通过所观测到的流量负载就能够自适应性地调整自身结构，从而达到最佳的性能。当负载处于可控范围时，正常情况下的客户端/服务器配置就能够应对用户的请求；而当负载超过某个阈值或满足特定条件时，代理缓存服务器集开始处理大量的请求。通过这种方式，网站在资源利用上更加高效，可更好地解决瞬时拥塞，同时多数 Web 站点也负担得起这样的配置方式。

在此方面，一般而言需要面临三个主要挑战：

1）如何快速组织一个代理缓存服务器的临时覆盖网。代理缓存服务器应当被有效地利用，并且在面临瞬时拥塞时需要及时彼此合作。当瞬时拥塞消失时，一切应当恢复原样，不需要增加其他系统的开销。在正常工作时，一台代理缓存服务器所带来的潜在影响应当被控制得尽可能小。

2）如何恰当地检测流量峰值的抵达和消失。瞬时拥塞与正常的工作负载不同，其幅值和延续时间取决于用户对某些突发性事件的兴趣所在，因此很难提前做出预测。也就是说，网络必须要通过短期快速预警来应对瞬时拥塞的出现。因此，对瞬时拥塞的检测必须做到小心谨慎，因为任何不恰当的检测都可能导致网络架构的剧变和资源的浪费。

3）如何以透明的方式将用户请求重定向到临时覆盖网中。针对瞬时拥塞的 CDN 动态架构一旦做好准备，海量用户请求就必须迅速被重定向到代理缓存服务器，并且

这些请求应当以一种负载均衡的方式被重定向。与单一站点机制不同（那里使用本地负载均衡系统），动态 CDN 的重定向必须在大尺度的临时架构环境中执行。

本章提出了 FCAN（瞬时拥塞缓解网络），这是一种动态 CDN，能够自适应地在 C/S 和 CDN 配置之间优化网络架构。我们使用了一种缓存代理服务器（cache proxy）的互联网设施来组织起代理缓存服务器之间的暂时性覆盖。当瞬时拥塞抵达时，该模式将被实时激活，而当正常情况下的 C/S 配置能够驾驭时，该模式将停止使用[7,33]。

本章的结构如下，11.2 节给出了瞬时拥塞的简介，并分析了其产生条件、类型和特性。除此之外，11.2 节还讨论了如何将瞬时拥塞与其他类似的互联网现象，如拒绝服务（DoS）攻击和分布式拒绝服务（DDoS）攻击进行区分，并介绍了该领域的研究现状，通过分析和比较若干相关的解决方案，列出了它们的优缺点。11.3 节提出了 FCAN，它规定了网络如何应对瞬时拥塞的开始和结束，以及如何组织代理缓存服务器及它们相互之间的临时协作，该节同时介绍了基于 DNS 的重定向技术如何帮助源 Web 站点分担负载，并给出了使用真实工作负载下的仿真结果。11.4 节总结了若干富有远见的观点；11.5 节给出了开放性问题和相关的未来研究方向；11.6 节进行了总结。

11.2 背景和相关工作

本节从研究瞬时拥塞的特性开始。我们的研究表明，网络带宽是最严重的瓶颈，并且绝大部分的请求仅仅涉及一小部分对象（也就是所谓的重尾行为特征）。这些现象表明，将大量请求通过重定向方式引离源服务器，并对那些导致瞬时拥塞的对象进行缓存，这可能是一种有效的解决方案。本领域的相关研究显示，目前处理瞬时拥塞问题的解决方案还存在着很大的潜力。

11.2.1 瞬时拥塞

一般而言，无论有意还是无意，公众感兴趣的突发事件会导致瞬时拥塞，著名的例子包括 1998 年世界杯[6]、RedHat Linux 镜像发布[8]、Play - along TV show 2000、1999 年的智利大选广播[21]以及 2001 年 9 月 11 日 CNN 对恐怖袭击的广播[13]等。除此之外，一个 Web 站点如果与某个流行内容、新闻或者博客相关，就会引发大量的突发访问。

因为资源和（或）网络带宽的限制，服务器经常无法处理海量请求。因此，大多数用户将明显感觉到网络性能的下降。另外，在发生瞬时拥塞时，与相关服务器共享带宽的其他用户将无法得到服务，同时也有可能殃及在该服务器检索其他信息的用户请求[30,51]。

通过以上的分析和其他研究结果[19,27]，我们能够得到一些关于瞬时拥塞的重要特性和结论（将在下面列出）。这些观测结果能够使我们预知瞬时拥塞何时抵达、需要采取相应行动的时间（多长或多短）、瞬时拥塞和恶意攻击的区别以及我们如何利用 Web 站点的本地性等。

1）瞬时拥塞发生时，虽然持续时间很短，但单位时间内请求量的增加是惊人的。瞬时拥塞取决于用户关注度的持续时间，从数小时到几天不等，与 Web 应用的生命周期相比较短。因此，如果使用过度配置的办法或转而使用传统 CDN，结果可能会导致在正常运行周期内资源的低效利用，尤其是对于小型网络或者私有的 Web 站点，因为这些小型站点在整个运营期内也不会遇见几次瞬时拥塞。

2）瞬时拥塞期间请求的增长速度很快，但非瞬时爆发。在 Play - along TV show 的例子中，请求率经过长达 15min 才达到顶峰。在另一个例子中，2001 年的 9・11 事件导致了 CNN 网站的巨大负载，访问数平均每 7min 翻一番，最终达到了 20 倍于日常负载的峰值[24]。这个特性表明，系统有足够的时间去检测到瞬时拥塞并做出及时的反应。

3）瞬时拥塞中网络带宽是一个主要的约束瓶颈。如果服务器提供动态生成内容的相关服务，那么 CPU 也可能成为性能瓶颈。例如，在 9・11 事件发生当天的上午，MSNBC 新闻网站动态页面出现的出错信息中，大约 49.4% 是代码为 500 的错误（服务器繁忙信号）[32]。然而，MSNBC 迅速切换到了静态 HTML 页面服务，使 500 错误的出现率降低到 6.7%。相关观测也表明，网络带宽是主要的瓶颈，而且离服务器越近所受影响越大[32]。有报道指出，现在桌面服务器能够在提供静态文件服务时达到 1Gbit/s 的网络吞吐量，而 Web 站点的网络带宽相对而言要小得多[40]，因此，我们应该重点关注如何减轻服务器周围的网络带宽瓶颈。

4）约有不到 10% 的内容占据了超过 90% 的访问请求。例如，MSNBC 的日志显示，在 9・11 当天超过 90% 的访问仅仅针对 141 个文件（0.37%），而 1086 个文件（2.87%）占据了 99% 的访问量[32]。另外，在瞬时拥塞中访问量大的内容一般不多，缓存完全可以装得下。这是一个重要结果，意味着针对 10% 左右内容的缓存大小可作为瞬时拥塞的解决方案。我们也观测到，这种“10/90”规则服从 Zipf - like 分布，其中一个针对第 i 个最受关注的内容的请求的相对概率与 $1/i^a$ 成正比[10]。这个特性将瞬时拥塞与由僵尸程序（bots）产生的攻击流量区分开来。

5）在瞬时拥塞期间超过 60% 的内容只被访问一次。另外，在上面所提及的 10% 的热点内容中，又有超过 60% 左右是新添加的缓存内容。例如，在 Paly - along 案例中 61% 的内容是未缓存的，在 Chile 案例中 82% 的内容是未缓存的[21]。这表明，一般的 Web 缓存难以提供系统期望的保护水平。大多数互联网上的缓存代理服务器（cache proxy）在瞬时拥塞开始时都不会存有用户所请求的内容，因此大多数请求都不会在缓存中得到服务，而是被转发给源服务器。尽管接下来的后继请求可能在缓存中能够得到服务，但大多数初始时的缓存缺失都会在开始很短一个时期内发生，并使请求被转发至源服务器。

6）在一次瞬时拥塞中用户的数量与单位时间内的请求率相同，这个特点使之区别于恶意请求。在一次瞬时拥塞中，请求流量的峰值对应于访问站点的用户数峰值，流量增加很大程度上源于用户数量的激增，而且大多数请求来自大量的用户群。然而，因为在瞬时拥塞中一台服务器的响应经常会慢下来，所以每用户请求率要比平时低，这表明是那些正常用户在影响着服务器的性能。

在研究瞬时拥塞行为时，我们需要区分某些相关但不同的现象，即 DoS 攻击。一个 DoS 攻击是一种“让目标主机停止向合法用户提供服务或资源访问”的攻击[12]，它通过海量数据包冲击目标主机，使其所有可用网络资源都被消耗殆尽，最后导致合法的用户请求无法进入，系统无法正常运行。一些常见的 DoS 攻击包括 SYN 攻击[11]、Code Red 攻击[29]、Password Cracking[18]等，还有就是最近常被提及的 DDoS 攻击[22]，这是一种通过大量被植入木马程序的受控计算机发送海量请求到目标的攻击。

DoS 攻击和瞬时拥塞有诸多相同点，它们都使服务器的互联网连接过载，并导致服务器部分或者完全失效。然而，服务器应当在处理瞬时拥塞时忽略 DoS 攻击，而仅处理正常用户的请求。

与瞬时拥塞相比，DoS 攻击具有以下明显特征[21]：①用户在 ISP 和网络中的分布并不服从总体分布（population distribution）；②在遭遇攻击前和被攻击中一个网站的用户群的重叠部分是非常小的；③用户请求率在攻击中是稳定的，并且与正常值偏离较大；④攻击的目标文件（未必存在于服务器上）的分布不可能是 Zipf - like 分布。

通过探寻这些不同，一台服务器有可能采取措施来区分 DoS 攻击和瞬时拥塞，并尽早丢弃那些恶意请求。服务器可以监控访问站点的用户以及他们的请求率，并对包的内容、HTTP 头和抵达率执行相关检查。

关于如何区别恶意请求和合法用户请求的更多细节和相关实现并不是本书要讨论的范围，详细资料可参见参考文献[21, 22, 35]。本章假定，服务器能够通过某些机制来滤除来自 DoS 攻击的恶意请求。

11.2.2 可行方案

根据网络的典型架构，目前用于解决瞬时拥塞的方法可以分为三大类：服务器层、中间层和客户端层，如图 11.2 所示。

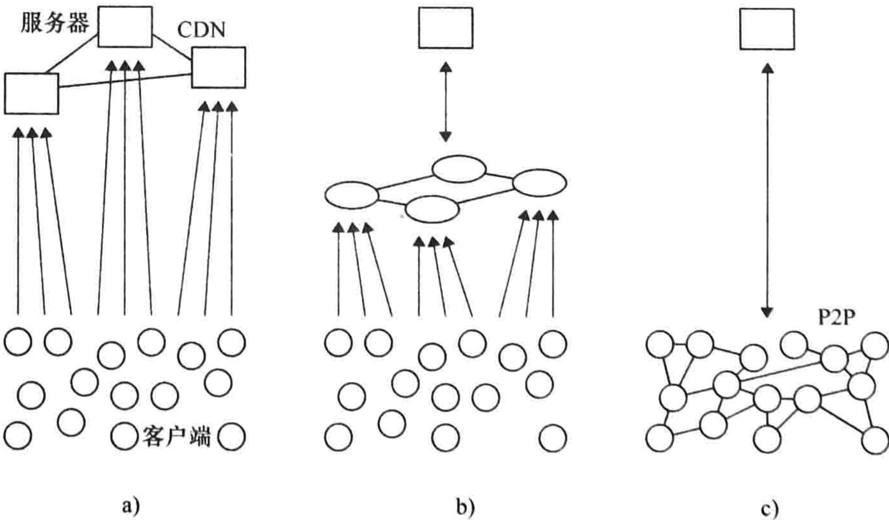


图 11.2　三种解决方案

a）服务器层　b）中间层　c）客户端层

1. 服务器层解决方案

如上所述，在服务器端使用过度配置和静态 CDN 的传统方法较为直接，但缺点是很耗资源[3, 25, 43]。此类方法性能低下，难以处理短期内的互联网拥塞现象。由于瞬时拥塞的不可预见性，任何不完备配置的系统都会在持续过载条件下出现问题。

（1）带有动态委托的 CDN

Jung 等人[21]提出了一种自适应的 CDN 结构，在过载情况下使用动态委托（dynamic delegation）对源服务器实现保护。该方法将代理缓存服务器（surrogate server）组织成不同的组，每一个组内会指定一台作为主服务器。一般情况下，CDN 中的 DNS 服务器会将用户请求只分配至主服务器。当主服务器的负载达到警戒值时，主服务器会请求 DNS 将请求分派至组内其他成员，这个被委托的服务器称为“代表”（delegate）。当一个代表收到一个缓存中缺失的请求时，它将该请求转发到主服务器而非源服务器上。这种机制称为“动态委托”，当代表参与进来后，系统的整体表现与协作式缓存相似。

如前所述，动态委托机制能够在瞬时拥塞初期处理 60% 左右的未缓存对象，并提高了系统的效率。它将高访问量的对象从源服务器中取出，通过分级缓存机制来消化造成缓存缺失的请求。然而，配备主服务器的代理缓存服务器组需要进行手动配置，并且一旦配置完毕便不能改变，甚至在正常运行期也是如此。

（2）DotSlash 方法

DotSlash 方法[50]允许不同的 Web 站点组成互助社区，并在社区内部使用空闲的服务器容量来缓解单一站点在瞬时拥塞期间的压力。作为一个救援系统，DotSlash 持续不断地监控每一台 Web 服务器的工作负载。当一台服务器过载时，救援服务立即被激活，而当服务器的负载回到正常值时，救援服务即停止。因此，Web 站点具备一个动态的服务器集，其中包含单个源服务器或者固定成员的源服务器构成的集群，以及一组数量可变的救援服务器。

与服务器层的其他方法及下面将要提到的中间层方法不同，DotSlash 方法并不使用一成不变的资源，它在一个临时组成的服务器互助社区中才激活其救援系统。然而，DotSlash 方法需要客户端和源服务器先建立连接，然后再指定一个经过重定向的 URL（包含有 DNS 重定向需要的虚拟主机名）。因此，此处可能存在风险，即在需要发送重定向报文时所需要的带宽和处理能力会使得源服务器失效。

2. 中间层解决方案

在应对瞬时拥塞的问题上已经有了若干中间层解决方案，这些方案大多使用网络资源来进行减负。缓存技术可以用于解决瞬时拥塞，通过过滤来自共享同一台代理服务器缓存（proxy cache）的用户组的重复请求来减轻服务器的负载。

一般而言，互联网上的代理服务器（proxy）可被分为两种：前向代理服务器和反向代理服务器。前向代理服务器被放置在客户端附近，因而远离源服务器。它们典型的功能配置包括防火墙和静态内容缓存。前向代理服务器通常被多数用户共享，因此需要性能强大和稳定可靠，不过内容提供商并不对其施加过多控制。反向代理服务器一般被放置在服务器集群的后端，作为应用提供商的代理（agent）使用，代表后

端服务器来响应用户请求。因此，内容提供商对其拥有完全的控制权。然而，反向代理的可拓展性只能由内容提供商的网络带宽决定[47]。

（1）多层缓存

多层缓存方法[4]认为，当使用某些恰当的替换算法时，设计出的用于处理正常Web负载的某种缓存设施也足以应对瞬时拥塞。该方案分别研究了使用不同缓存替换算法、改变缓存放置地点、使用异质性多层缓存以及将基于文档大小的ID空间分区的效果。研究结果表明，通过使用GDSF算法[5]，缓存中的替换策略能够对用户响应时间、服务器和网络负载产生非常明显的改善。

多层缓存技术为使用缓存来解决小型、静态对象的瞬时拥塞问题提供了很好的解决方案。此系统需要对缓存进行精细的分层部署，并且需要对前向缓存代理服务器的设施架构拥有完全控制权。此系统并没有解决60%未缓存对象的问题，因此可能无法提供对源服务器更好的初期保护。另外，此类技术目前仍缺乏用于灵活处理瞬时拥塞的自适应机制。

（2）BackSlash

BackSlash[44]技术使用基于分布式哈希表（DHT）的内容可寻址的P2P覆盖技术来建立分布式Web服务器系统。它将内容的副本存放于由内容提供商指定的镜像服务器。DHT提供了参与者自组织、请求路由和负载均衡的基础。

BackSlash方法使用Web服务器和代理服务器来对网络流量进行分流。然而，在BackSlash中镜像服务器上的内容必须提前配置好并且进行妥善组织，这增加了运行的复杂度并限制了系统的可拓展性。

（3）CoralCDN

CoralCDN[17]利用志愿者的带宽对大多数使用该系统的Web站点流量进行消化和分流。在第1章已经提到，CoralCDN在“键值/索引”架构上使用了覆盖路由技术，即一个P2P的分布式散列哈希表（DSHT）能够允许节点对附近Web对象的缓存副本进行定位，而无须查询远程节点，这也同时避免了架构中热点的出现，即使在负载恶化时也是如此。为了使用CoralCDN，内容发布者或提供连接到高流量端口的管理员需要将URL的主机名后附加“. nyud. net：8090”。

Coral使用志愿者的带宽资源来消化巨大的流量，它将一组基于P2P的反向代理服务器合并起来，创建按需提供的缓存对象，并使DNS以透明方式重定向用户请求。CoralCDN总是等待请求的进入，请求的URL需要手动添加“. nyud. net：8090”。URL经过修改之后，CoralCDN即能进行面向对象的重定向，然而它牺牲了系统对用户的透明性。

3. 客户端层解决方案

客户端层解决方案通过利用客户端之间的协作和共享对象来达到减轻中心服务器负载的目的。一台Web源服务器可以将一个客户端重定向到近期内下载过所需对象的另一个客户端来实现客户端之间的协作，如Squirrel[28]、Pseudoserving[23]和Coop-Net[32]。客户端也可以通过组成P2P覆盖网并使用搜索机制来定位资源。例如，PROOFS[45]利用随机化的方法来建立客户端的P2P覆盖网，BitTorrent[9]将大文件分割

成小块便于有效检索。一般而言，这些方法依赖于客户端的协作，它们需要被安装在客户端的计算机上，这样才有可能避免花费力气在大范围内的部署工作。

（1）CoopNet

CoopNet（协作网）[32]是一个 P2P 缓存解决方案，用于客户端与服务器以及客户端与 Web 代理服务器之间的通信服务。在 CoopNet 中会存在已经下载过内容的注册用户，并且这些用户反过来又可将内容传输给其他用户。CoopNet 使用基于 HTTP 的重定向技术来路由请求，并根据请求附近的定位信息来选择其对等成员。

在 CoopNet 中，P2P 通信在瞬时拥塞出现时被激活并分担负载，在 C/S 通信工作正常时停止工作。CoopNet 使用基于服务器的重定向会导致单点失效的风险。

（2）PROOFS

PROOFS[39, 45]由两个协议组成：第一个构成并维护网络覆盖；第二个在第一层形成的覆盖模式上执行一系列随机的广域对象搜索。节点持续执行一种被称为“洗牌策略”的交换模式。所谓洗牌策略，就是在客户端对之间的邻子集中的彼此交换，并可通过任意客户端进行初始化。洗牌的目的是形成一个充分混合的覆盖，因为一个客户端的邻居是通过随机方式从所有参与到覆盖的客户端集中提取出来的。一旦达到一个随机状态，广域对象搜索便在覆盖网络上开始进行。对象随机访问邻居集，直到找到一个包含该对象的节点。通过理论分析和仿真实验，PROOFS 据称能够在覆盖分区、对等成员动态加入/离开系统及限制参与系统等方面表现出很好的鲁棒性。

PROOFS 使用了非结构化的第一代 P2P 系统，因此配置简单且成本低廉，并在面对瞬时拥塞时具有良好的性能。现在研究重点都集中在第二代 P2P 系统架构上，如 CAN[36]、CHORD[46]和 Pastry[38]，在这些系统中参与者比较清楚请求应该如何转发。他们能够提供第一代系统所不具备的带宽使用优势和定位文件所需的时间优势。不过，如果想处理好那些访问量突然激增的文档，同时还要保护好提供这些文档服务的服务器节点，那么第一代架构（它们更简单且更轻量级）则是理想的选择。

4. 其他工作

网格技术能够解决在动态的多组织机构间彼此协作时的资源共享问题[16]，它主要关注大尺度的计算问题以及包括多用户和不同类型的行为与交互等的复杂应用问题。互联网数据中心利用共享的硬件资源托管多个 Web 应用。Chandra 等人通过将资源重新分配到过载应用来动态改变应用的负载，也就是说，在必要时从那些资源并未得到充分利用的应用中“借”资源[13]。

在另一个方法中，一个 Web 站点可以使用访问控制[14, 15, 49]，通过拒绝一部分用户请求并有选择性地服务特定用户来避免过载。

11.3 FCAN：瞬时拥塞缓解网络

FCAN[7, 33]是一个中间层解决方案，它使用类似于 CDN 的缓存代理服务器的大范围覆盖网络来存储对象，并将对象传递到客户端，这与 CDN 中的代理缓存服务器类似。考虑到瞬时拥塞的不可预见性以及较短的持续时间，FCAN 仅当一台服务器过载

时才激活覆盖架构，并在必要时重新组织覆盖。与上文已提及的大部分方法相比，这种动态性是 FCAN 最重要的特性。唯一的例外是 DotSlash，不过它缺乏自适应的重组织特性。

FCAN 的目的在于完善现有的 Web 服务器设施，以便能有效应对短期性的负载暴增。FCAN 目的并不是为那些长期超过 Web 站点负载能力的请求而设计的，因此它针对的仅仅是小型的 Web 站点，虽然大型 Web 站点也能通过 FCAN 实现有效运转。

11.3.1 需求

下面列出让 FCAN 能有效、可靠及经济运行所需的功能性和非功能性的系统需求。

（1）对象分发

第一位也是最重要目的是内容对象的及时发布。FCAN 应在任何时候都能维护内容分发服务的高可靠性。另外，FCAN 也应该确保对服务器上非瞬时拥塞对象的可访问性。

（2）工作负载控制

FCAN 应该能够监控负载增加的情况并对其进行控制，使服务器不至于过载。同时，当海量请求被临时性的代理缓存服务器分流时，FCAN 应当在每一台代理缓存服务器上对工作负载进行监控，并检测瞬时拥塞现象的消失和控制重定向的请求，使代理缓存服务器不至于出现过载的情况。

（3）自适应转换

FCAN 应当能够对负载增加的情况保持足够的敏感，并能够以自适应的灵活方式来改变自身的架构，以获得最佳性能输出。由于架构的转换时间和持续时间应当尽量短，所以检测和转换都应当以自动方式完成。

（4）请求重定向

FCAN 需要通过恰当的机制找到临时的代理缓存服务器并重定向海量请求。不仅如此，FCAN 也要能选择最合适的代理缓存服务器。如果执行的重定向能够在全局上实现均衡，这种机制就是理想的。

（5）客户端透明度

如果客户端无需改变，那么 FCAN 将更容易被客户接受。因此，对于客户端而言，如果他们在使用过程中完全没有意识到 FCAN 的存在，那么 FCAN 的目的就达到了。

（6）可拓展性

因为基于互联网的设施具有沟通全球的特性，所以 FCAN 要能够在最小代价下做到易于扩展。

11.3.2 设计总览

在正常情况下，传统的 C/S 架构能够处理大部分客户端请求。构成 FCAN 的一个成员服务器和成员缓存代理服务器在正常情况下并不比普通服务器优越很多。当瞬时

拥塞抵达时，成员服务器检测到了流量负载的激增，它随即激活部分成员代理服务器形成一个覆盖（overlay），从而使所有的请求得到处理。随后的所有客户端请求都通过 DNS 重定向路由到这个覆盖。如果这些代理服务器还不足以处理海量请求，新的代理服务器会加入，使覆盖扩大。当瞬时拥塞消失时，部分代理服务器会离开覆盖，这时覆盖开始减小规模，直至最终解散。图 11.3 给出了 FCAN 的三种不同的工作状态，分别为常态、初态和扩张态。

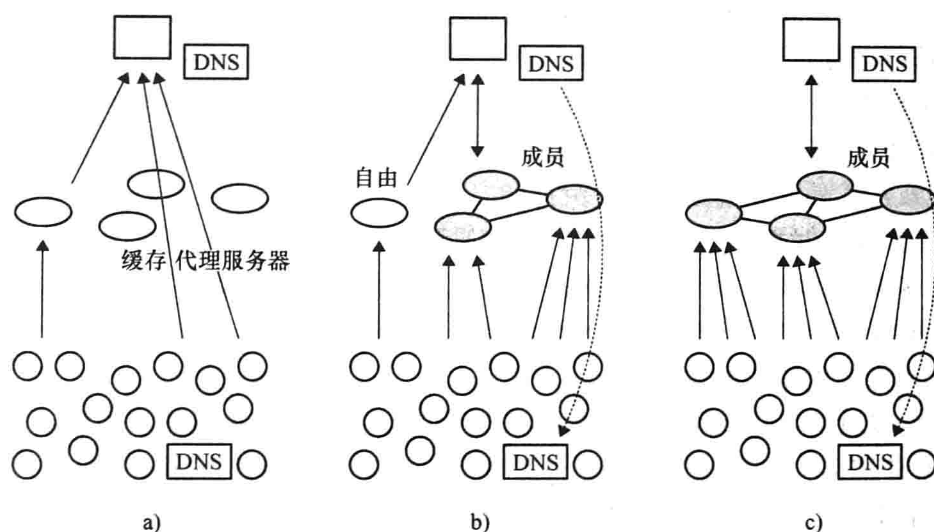


图 11.3　FCAN 概况
a）常态　b）初态　c）扩张态

FCAN 并不专为某一单独成员服务器使用，而是设计为由多台服务器在需要时共享。每一个成员服务器都使用自己的覆盖，不论该覆盖大还是小。服务器之间会尽量相互避免重叠过多。

在一般状态下，每一台成员代理服务器通常主要充当一台转发缓存代理服务器。不过，它充当的是代理缓存服务器，多少有点类似于反向缓存代理服务器（reverse cache proxy），在“反瞬时拥塞”模式下处理任意客户端的请求。在现实中，一个成员代理服务器为多台服务器服务，这时可能存在一种情况，即某台服务器受到瞬时拥塞的影响而其他的却没有，因此每一个成员代理服务器都有混合模式运行的功能（即同时具备前向代理模式和反向代理模式）。

11.3.3　瞬时拥塞检测

网络带宽、服务器 CPU 和内存等不同资源都有可能在瞬时拥塞中成为瓶颈，不同的资源在测量工作负载时应该采用不同的指标。每一个成员服务器都应完成监控和过载/欠载检测，并相应地执行动态转换功能。现有的设计仅仅使用访问到达率来作为负载的衡量准则，并使用基于阈值的技术来激活动态转换。

为了检测瞬时拥塞的到来，服务器需要周期性地观测自身的负载。一旦服务器负载超过预定阈值 T_{high}，服务器就可以确定瞬时拥塞的出现。

在瞬时拥塞期间，覆盖中的每一个代理服务器都有一个负载监控系统，负责观测用户访问的数量。成员服务器从所有相关的代理服务器中定期收集信息，如果这个覆盖内的负载低于预定阈值 T_{low}（$<T_{high}$），那么成员服务器判断此时瞬时拥塞结束。

11.3.4 网络转换

当成员服务器检测到瞬时拥塞时，它将执行如下步骤，使某些成员缓存代理服务器切换到反瞬时拥塞模式，并组成临时的覆盖网：

1）选择部分代理服务器，形成 CDN 那样的代理缓存服务器覆盖网。

2）进行 DNS 记录的更新，改变 Web 站点的检索项（也就是从服务器地址到代理服务器的映射），这样一来，后继请求将在 DNS 的帮助下被重定向到适当的代理服务器。

3）将涉及瞬时拥塞的对象传播（推送）到选定的代理服务器，因为上文已经提到，超过 60% 的瞬时拥塞对象在海量访问抵达之前都是未经缓存的。

4）准备收集相应代理服务器上的对象信息并评估对象的统计特性，目的是确定动态重组和将来覆盖的释放。

每一个成员缓存代理服务器在收到成员服务器的请求后执行如下步骤：

1）改变当前自身模式，从代理服务器模式切换到代理缓存服务器模式（或者从严格意义上来说，如前文所述包括前向代理和反向代理的混合模式）。

2）永久性存储产生瞬时拥塞的对象，直到瞬时拥塞消失。

3）开始监控请求率和负载的统计数据，并周期性向服务器报告。

服务器首先选择一部分代理服务器，对它们逐个进行检测，这是因为任意代理服务器都有可能正涉及另一个瞬时拥塞，或者有可能由于其他原因处于过载状况。这样做的目的是在面临多个不同瞬时拥塞时，避免相应的代理服务器子集相互重叠。因此，最终选择出来的代理服务器子集可能较小，甚至有可能仅含一台代理服务器，这源于 FCAN 具备动态重组的特性，后面将详细介绍。现有设计模式需要网络管理员在代理服务器进行探测时为探测顺序分配一定的优先级。

当成员服务器检测到瞬时拥塞消失时，所关联到的所有代理服务器子集会全部解散，顺序采用逆序的方式（与开始探测时的排序方式相反）。执行的步骤为：

1）服务器更新 DNS 记录；

2）服务器通知代理服务器解散；

3）代理服务器改变自身模式，从代理缓存服务器模式变成代理服务器模式。

当所有的代理服务器解散后，这种类似 CDN 的覆盖网转换回正常的 C/S 模式。注意，服务器不是立刻解散的，因为低负载仅仅是暂时的，因此系统此时还应处于反瞬时拥塞模式。

11.3.5 动态重组

每一台代理服务器都有自己的本地监控系统，可以检测用户请求率和自身的整体负载。覆盖中的代理服务器无论初始时就存在还是后来加入，都周期性发送反馈信息

到服务器，该反馈信息包括瞬时拥塞对象的请求率和代理服务器上的整体负载情况。

当请求率逼近 T_{high} 时，代理服务器将服务器请求率已经逼近临界点并还在增加的情况上报。当大多数代理服务器都上报同样信息时，服务器开始加入更多的代理服务器。这些被加入的代理服务器处于空闲池中，并且没有加入任何覆盖。此时，服务器对空闲的代理服务器逐个进行探测，并选出一个开始使用。随后，服务器与这台代理服务器一同执行 11.3.4 节中描述的步骤。

当任意一台代理服务器上的负载低于 T_{low}，并且其他的代理服务器未处于高负载情况时，系统将它们解散，解散的顺序与代理服务器的加入顺序相反。因为 DNS 的传播需要花费较长的时间，因此代理服务器运行模式的改变要稍晚一些。如果在 DNS 更新之后客户端仍然连接到这个代理服务器，那么该服务器将充当通常的前向代理服务器，它将从本地缓存中对内容进行检索，或将用户的请求重定向到其他成员服务器或覆盖中的任意一台临时代理缓存服务器。

图 11.4 给出了服务器的工作流程图、初始和辅助代理服务器以及网络转换（见 11.3.4 节）和动态重组。

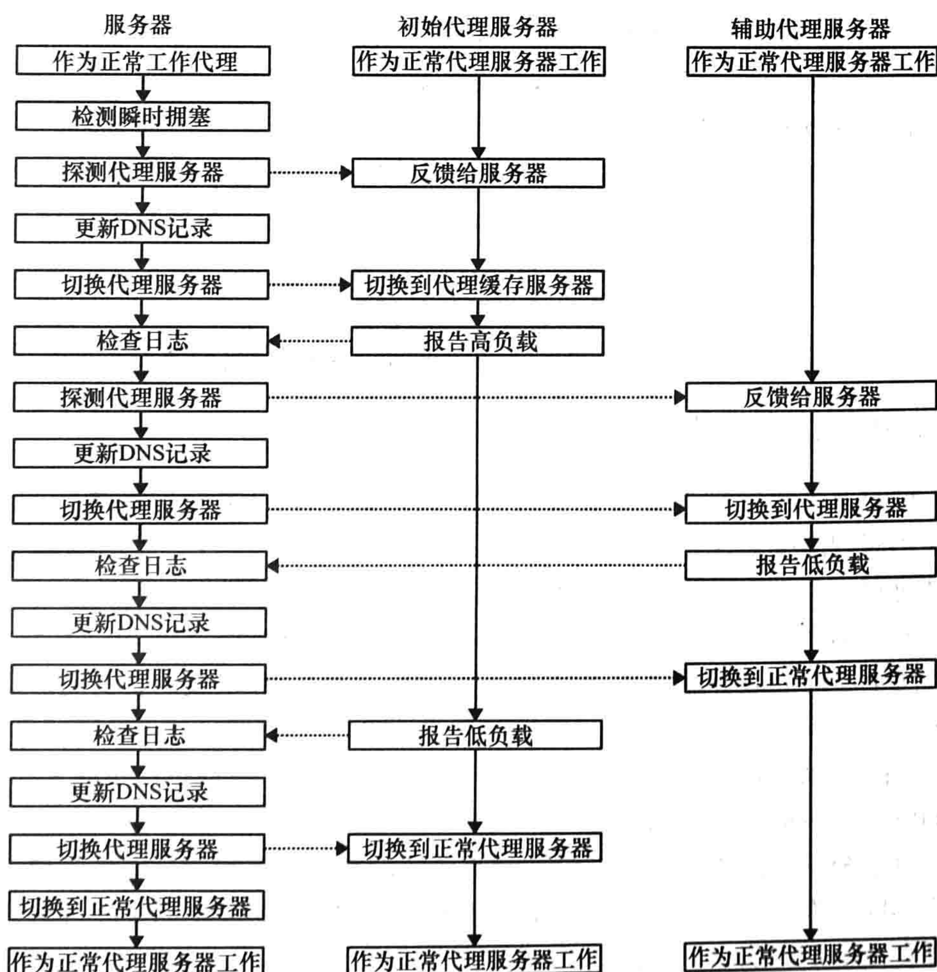


图 11.4　FCAN 的工作流程图

11.3.6 基于 DNS 的重定向

为了避免服务器和网络的过载，必须对海量请求进行重定向。与单一站点机制中本地负载均衡系统不同，这个重定向是在大范围环境中完成的，其中代理服务器之间的距离可能很远。上文已经提及，我们使用了基于 DNS 的请求重定向技术。DNS 是所有互联网应用中都需要的设施，在互联网上无处不在，并且对于客户端来说是透明的。

当一个客户端试图通过本地 DNS 服务器解析服务器域名时，成员服务器的 DNS 输出的是相应的成员缓存代理服务器的地址而非源服务器的地址。提供给客户端的地址可能是通过某一选择策略选择出来的任一代理服务器，选择策略可以是通过一种简单的轮询方式，也可能是以负载均衡方式或者基于最近邻的方式。经过重定向的海量请求由目标代理服务器来进行处理[48]。

我们使用一台特殊配置的 DNS 服务器（或者从严格意义上说，DNS 封装器），称为 TENBIN[41,42]，它运行在服务器端，允许 DNS 中的映射项被动态修改。TENBIN 是我们的一个研究成果，并且已经用于实际应用，如“Ring Server”[37]和“LIVE! ECLIPSE”[26]。TENBIN 也支持用于选择“合适”地址的策略配置。该策略可以基于负载权重算法、最近邻的算法、缓存本地性的算法或简单的轮询算法。

一旦地址被修改，新地址便通过互联网传递至客户端的 DNS 服务器。一个问题是，DNS 缓存可能会造成传播过程的延时，其结果是用户请求仍然需要发至源服务器。为了解决这个问题，可以设置 DNS 中的记录，赋予其一个较短的失效时间，即零生存周期（zero TTL）。我们已通过 TENBIN 的实验和实际应用积累了大量关于 DNS 传播的经验数据。一般而言，完成全球范围的传播需要花费 10 ~ 15min 时间，但是与可能长达数小时或数天的瞬时拥塞相比，这个时间几乎可以忽略不计[12]。

11.3.7 基于仿真的性能评估

为了对 FCAN 进行初步的验证和评估，我们建立了一个基于线程的包含 TCP/UDP 和应用层的虚拟网络用于仿真实验。我们运行了若干实验，分别测试了不同类型的瞬时拥塞。下面给出其中的一个实验，它提供了真实的访问日志数据，该数据来自“LIVE! ECLIPSE”项目[26]。在 2006 年 3 月 29 日，从 GMT 时间 9：00 到 11：30，该项目从两个不同的服务器站点分发来自土耳其、利比亚和埃及的日食流信息。这两个站点是：

- http：//www. live - eclipse. org
- http：//www. nishoku. jp

前者对于全球用户均可访问，后者的主要访问者为日本用户。在访问模式上两个站点有所不同，因为前者的平均访问率要比 Nishoku 站点高得多。图 11.5 给出了两个站点在日食发生期间的访问日志。

在将数据送入模拟器时，站点的日志数据被按比例缩小。其中，Live - Eclipse 站点的数据被缩小到 1/30，而 Nishoku 缩小到 1/10。在仿真中每秒对应于真实时间的

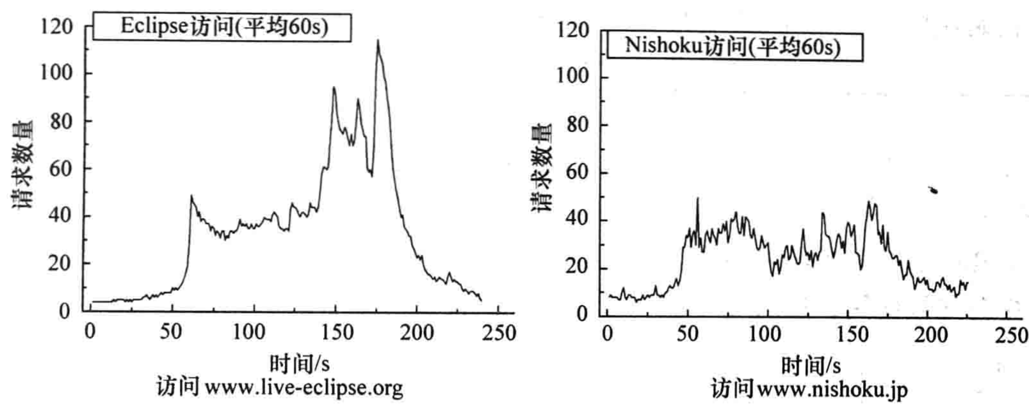


图 11.5　对两个日食流站点的访问

1min。实验中使用了两台不同的成员服务器：用于 Live - Eclipse 的 SVR01 和用于 Nishoku 的 SVR02。实验中使用了 10 台成员缓存代理服务器用于缓解瞬时拥塞。为这些成员服务器设置的代理服务器优先级（即探测顺序）是不同的，并且代理服务器的初始子集合也不相同，这些不同的设置主要是根据成员服务器的优先级和瞬时拥塞的强度。

图 11.6 和图 11.7 给出了仿真结果。其中，图 11.6 给出的是在 SVR01 周围产生的服务于“Live Eclipse”的覆盖，图 11.7 给出的是对应于“Nishoku”的 SVR02 覆盖。在这两个图中，左图显示的是服务器的平均负载，右图是个体负载。

在“Live Eclipse”覆盖中有 7 台代理服务器，其中两个是初始代理服务器，五台是辅助代理服务器，如表 11.1 所示。

表 11.1　Live Eclipse 覆盖中的代理服务器

SVR01	加入时间	离开时间
CP8（初始）	63	211
CP3（初始）	63	211
CP2	65	191
CP5	145	190
CP7	149	189
CP1	155	188
CP9	174	184

在最初的 60s 中，服务器 SVR01 单独处理用户请求。当瞬时拥塞在第 60s 左右到达时，服务器加入两台代理服务器以缓解瞬时拥塞，由这两台代理服务器和另外一台随后加入的辅助代理服务器对负载进行处理。在第 150s 左右当负载的第二个快速增长发生时，又有四台代理服务器被加入。当使用所有的 7 台代理服务器时，可以发现平均负载已经被保持在了阈值以下。在第 180s 处，用户请求数开始下降，系统随之开始逐步减少代理服务器数量，直到系统切换回正常的 C/S 模式，这一切换大约在第 200s 发生。

在“Nishoku”覆盖中，因为相对负载较低，所以只加入了两个代理服务器，即

一台初始代理服务器和一台辅助代理服务器，如表 11.2 所示。

表 11.2 Nishoku 覆盖中的代理服务器

SVR02	加入时间	离开时间
CP0（初始）	48	189
CP6	49	51

SVR02 的瞬时负载峰值在第 50s 左右出现，此时已达到用户请求的峰值，CP0 在覆盖中首先被使用，然后 CP6 立刻被加入，但仅仅持续了 2s。

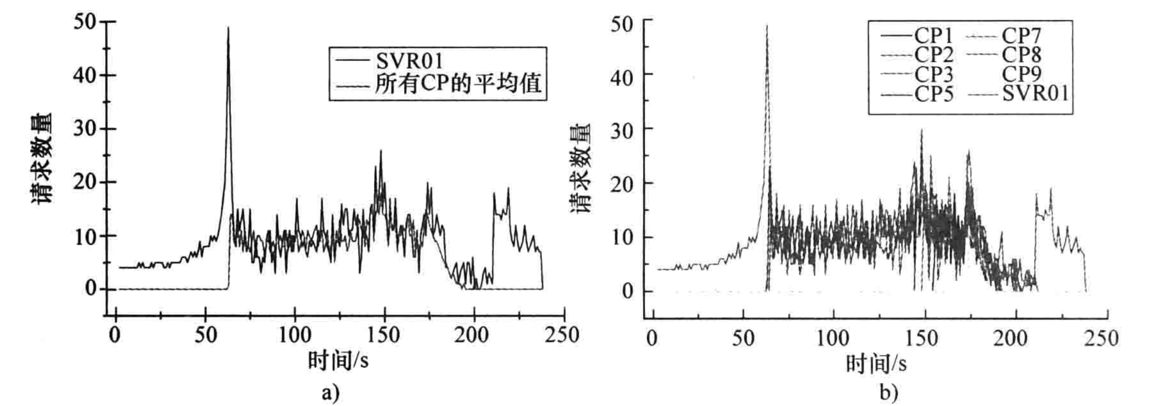


图 11.6 Live Eclipse 覆盖中的负载缓解情况

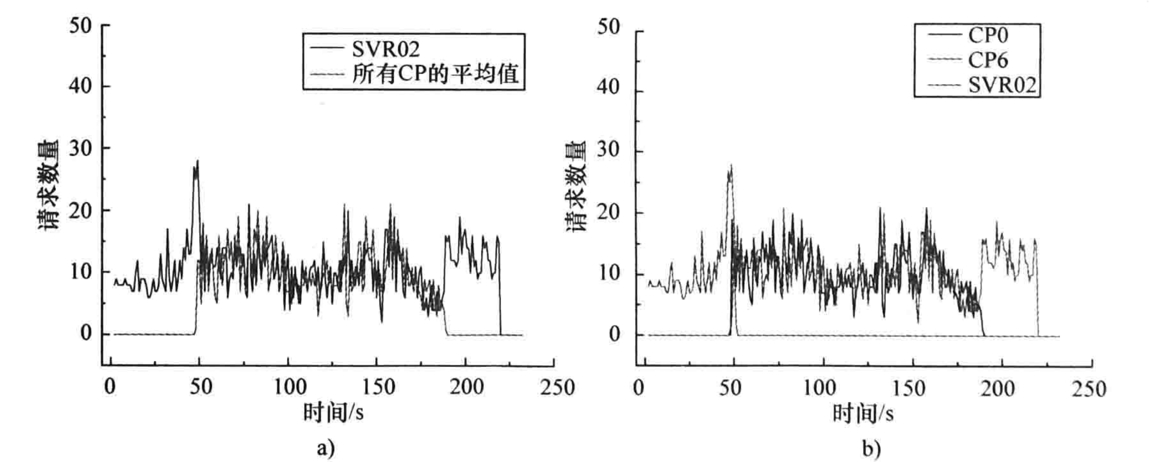


图 11.7 Nishoku 覆盖中的负载缓解情况

11.3.8 结论

FCAN 最突出的特性在于它的动态性、自适应组织能力以及类似于 CDN 覆盖的重组特性，据我们所知，目前尚没有其他工作具备这些特性。我们提供的相关数据显示了 FCAN 的这些特性所带来的功效。与基于客户端层的解决方案相比，使用缓存代理服务器的中间层解决方案更易于管理和控制。另外，与基于服务器层的解决方案相比，FCAN 更灵活，也更贴近用户。

现有的 FCAN 设计仅仅是第一版，因此它仍有很多需要改进之处。基于阈值的瞬时拥塞检测方法会变得更复杂些，这一点将在后文提及。基于优先级的代理服务器分组机制现在正在被一种自治的非中心化聚类法取代。

另外一个要点是重定向的粗粒度问题。瞬时拥塞是面向对象的，而基于 DNS 的重定向是面向主机的，因为 DNS 仅仅处理主机名。因此，较为可行的做法是仅把瞬时拥塞的对象定向到代理服务器覆盖，而其他针对非瞬时拥塞对象的请求则像正常情况一样直接处理。基于 HTTP 的重定向和 URL 静态化技术也提供了精细粒度的面向对象的重定向，但是这些方法对用户并不透明。

目前为止，本章所设计的初步仿真并不包括对 FCAN 的定量分析和全面评估。在互联网真实环境中，FCAN 应当由如下部分组成：

1）一台专用的 Web 服务器。该服务器配备一个实现 FCAN 功能的封装器模块，其核心可以是 Apache，封装器应当拦截前往核心服务器的请求。

2）一台专用的缓存代理服务器。该代理服务器配备一个实现 FCAN 功能的封装器模块。其核心可以是 Squid，封装器应当拦截前往核心代理服务器的请求。

3）一台增强的 DNS 服务器。TENBIN 是一个很好的选择，事实上 TENBIN 已经得到了实际应用。

FCAN 最初的设计目标是提供瞬时拥塞情况下的保护，但事实上它的用途并不仅限于此。在任何突发的流量激增时，它可以在预定阈值下对服务器负载进行调整，因此它也能应对某些 DDoS 攻击。

11.4 写给使用者

表 11.3 总结了若干相关研究的重要成果（它们中的大多数已经在本章中提及），并把它们和 FCAN 进行了比较，我们的结论如下：

表 11.3　相关系统应用的设计汇总表

设计问题		DD	DS	ML	BS	CO	CP	PF	FC
系统架构	服务器层	√	√						
	代理服务器层			√	√	√			√
	客户端层						√	√	
代理缓存服务器	指定服务器	√							
	已有服务器		√		√				
	已有代理服务器			√	√	√			√
	已有客户端			√			√	√	
客户端透明度	客户端无意识	√		√					√
	浏览器不改变	√	√		√	√		√	√
客户端重定向	基于 DNS	√	√		√	√			√
	URL 改写	√	√		√	√			
	基于 HTTP						√		
副本放置	镜像副本	√	√						
	按需缓存	√	√	√		√		√	√

（续）

设计问题		DD	DS	ML	BS	CO	CP	PF	FC
对象定位	基于 DHT 的 P2P				√	√			
	非结构化的 P2P						√		
	协作缓存			√			√	√	√
缓存缺失避免	动态委托	√							
	推送服务								√
自适应转换	临时服务器		√						
	临时代理								√
	临时客户端						√		

注：DD：带有动态委托的 CDN，ML：多层缓存，BS：BackSlash，DS：Dot. Slash，CO：CoralCDN，CP：CoopNet，PF：PROOFS，FC：FCAN。

1）基于需求峰值的过度配置或者使用 CDN 来预先提高服务器定位的方法代价高昂且低效。

2）客户端的 P2P 覆盖可以很好地解决瞬时拥塞问题，但是解决得并不完美，因为它失去了对客户端的透明性和可控性。

3）另外，某些 P2P 系统也面临流量超限等问题，这些问题在瞬时拥塞消失时不能忽略。

4）中间层解决方案相对其他解决方案而言具有一定优势，这是因为缓存技术在解决目标对象为静态且尺寸较小的瞬时拥塞问题时非常有效。

5）大多数中间层解决方案忽视了一个问题，即 60% 的对象在瞬时拥塞开始时并没有被缓存，这导致源服务器会面临大量缓存缺失的风险。

6）相对于服务器，前向代理服务器更适合被部署为代理缓存服务器。代理服务器离用户更近，因此对改善用户响应时间和网络拥塞更有效。

7）为了灵活和高效地处理瞬时拥塞问题，自适应的转换技术必不可少，该技术能够随时根据需要组织潜在资源，而不是一直占据它们。

总体而言，现有研究成果都各具优缺点。通过比较，我们得出的结论是，目前仍然缺少一种高效的方法来以透明的方式灵活、可靠和高效地解决瞬时拥塞问题。

11.5 未来研究方向

虽然目前在缓解瞬时拥塞方面有很多问题需要解决，但我们将重点关注以下两个问题。

（1）瞬时拥塞的早期检测

我们已经注意到一种现象，即在瞬时拥塞到达服务器之前很短一段时间内，大量请求有时会涌向 DNS 服务器去解析服务器域名。这表明，如果有一种技术能够从分布于各处的 DNS 服务器上收集到请求的数量并进行分析，就可以预测到瞬时拥塞的来临，并能够对目标服务器发出早期预警信号。

（2）处理动态对象

对于所有的 CDN 系统而言，实现动态对象的发布必须是一项基本功能。动态对象可分为以下两类：

1）动态生成的内容（大多数使用脚本代码和后端数据库）；

2）频繁更新的内容（常见于新闻网站）。

最简单的方法就是在过载情况下将动态对象用修改过的静态版本替换，其代价是牺牲了服务质量[1]。

如果后端数据库是只读的，那么处理第一种动态对象一定会相对容易。如果后端数据库并非只读或者动态对象属于第二类，那么就必须提供一种快速和可靠的机制来同步地更新所有副本。这个保证更新同步化和一致性的问题已经在分布式数据库领域、分布式缓存和分布式共享内存等领域中进行了广泛和深入的研究，其研究成果均可应用于本章所讨论的问题中。

最后，一些集成了服务器层、中间层和客户端层的综合解决方案也有一定的发展前途，值得引起读者和相关研究人员的关注。

11.6 总结

短时互联网拥塞（即瞬时拥塞问题）对于设计可扩展和高效的分布式服务器系统提出了新的挑战。本章分析了瞬时拥塞的主要特性，针对相关研究方向进行了介绍，并指出了处理短时互联网拥塞的动态网络的必要性。我们原创性地提出了一种新型的动态 CDN，能够使自身网络架构在 C/S 和 CDN 之间进行自适应优化，由此减轻瞬时拥塞带来的影响。

仿真结果表明，FCAN 能够为瞬时拥塞的早期检测提供良好的基础，并能够处理动态对象。因此，我们的结论是：FCAN 能够为高效处理瞬时拥塞问题提供一条未来网络革新的重要思路。

致谢

本章内容来自 Kyushu Sangyo 大学的 Toshihiko Shimokawa 教授、中国的 Chenyu Pan 博士、土库曼斯坦的 Merdan Atajanov 博士的合作，同时本章作者也对 Kate Miller 女士的校对工作表示感谢。

参考文献

[1] Abdelzaher TF, Bhatti N (1999) Web Server QoS Management by Adaptive Content Delivery. In: Computer Networks, 31(11–16):1563–1577

[2] Adler S (1999) The Slashdot Effect, an Analysis of Three Internet Publications. http://ssadler.phy.bnl.gov/adler/SDE/SlashDotEffect.html

[3] Akamai Technologies Inc. http://www.akamai.com

[4] Ari I, Hong B, Miller EL, Brandt SA, Long DE (2003) Managing Flash Crowds on the Internet. In: Proc. 11th IEEE/ACM Int. Symp. on Modeling, Analysis, and Simulation of Comp. and Telecomm. Sys., 246–249

[5] Arlitt M, Cherkasova L, Dilley J, Friedrich R, Jin T (1999) Evaluating Content Management Techniques for Web Proxy Caches. In: ACM SIGMETRICS Performance Evaluation Review, 27(4):3–11

[6] Arlitt M, Jin T (2000) A Workload Characterization of the 1998 World Cup Web Site. In: IEEE Network, 14(3):30–37

[7] Atajanov M, Shimokawa T, Yoshida N (2007) Autonomic Multi-Server Distribution in Flash Crowds Alleviation Network. In: Proc. IFIP 3rd Int. Symp. on Network Centric Ubiquitous Systems (LNCS 4809, Springer), 309–320
[8] Barford P, Plonka D (2001) Characteristics of Network Traffic Flow Anomalies. In: Proc. ACM SIGCOMM Internet Measurement Workshop, 69–73
[9] BitTorrent Website. http://www.bittorrent.com/
[10] Breslau L, Cue P, Fan L, Phillips G, Shenker S (1999) Web Caching and Zipf-like Distributions: Evidence and Implications. In: Proc INFOCOM 1999, 126–134
[11] CERT (1996) TCP SYN Flooding and IP Spoofing Attacks. Advisory CA-1996-21, http://www.cert.org/advisories/CA-1996-21.html
[12] CERT (1999) Denial of Service Attacks. http://www.cert.org/tech_tips/denial_of_service.html
[13] Chandra A, Shenoy P (2003) Effectiveness of Dynamic Resource Allocation for Handling Internet Flash Crowds. Tech. Report, TR03-37, Dept. of Computer Science, Univ. of Massachusetts Amherst
[14] Chen X, Heidemann J (2002) Flash Crowd Mitigation via an Adaptive Admission Control Based on Application-Level Measurement. Tech. Report, ISI-TR-557, USC/ISI
[15] Cherkasova L, Phaal P (2002) Session-Based Admission Control: A Mechanism for Peak Load Management of Commercial Web Sites. In: IEEE Trans. on Computers, 51(6):669–685
[16] Foster I, Kesselman C, Tuecke S (2001) The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In: Int. J. of High Performance Computing Applications, 15(3):200–222
[17] Freedman MJ, Freudenthal E, Mazieres D (2004) Democratizing Content Publication with Coral. In: Proc. 1st USENIX/ACM Symp. on Networked Systems Design and Implementation
[18] Houle KJ, Weaver GM, Long N, Thomas R (2001) Trends in Denial of Service Attack Technology. CERT Coordination Center White Paper, http://www.cert.org/archive/pdf/DoS trends.pdf
[19] Iyengar AK, Squillante MS, Zhang L (1999) Analysis and Characterization of Large-Scale Web Server Access Patterns and Performance. In: World Wide Web, 2(1–2):85–100
[20] Joubert P, King R, Neves R, Russinovich M, Tracey J (2001) High-Performance Memory-Based Web Servers: Kernel and User-Space Performance. In: Proc. USENIX 2001, 175–188
[21] Jung J, Krishnamurthy B, Rabinovich M (2002) Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In: Proc. 11th Int. World Wide Web Conf., 252–262
[22] Kandula S, Katabi D, Jacob M, Berger A (2005) Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds. In: Proc. USENIX 2nd Symp. on Networked Systems Design and Implementation, 287–300
[23] Kong K, Ghosal D (1999) Mitigating Server-Side Congestion in the Internet through Pseudoserving. In: IEEE/ACM Trans. on Networking, 7(4):530–544
[24] LeFebvre W (2002) CNN.com: Facing a World Crisis. In: USENIX Annual Tech. Conf., http://tcsa.org/lisa2001/cnn.txt
[25] LimeLight Networks. http://www.limelightnetworks.com/
[26] LIVE! ECLIPSE. http://www.live-eclipse.org/index_e.html
[27] Lorenz S (2000) Is Your Web Site Ready for the Flash Crowd? In: Sun Server Magazine 2000/11, http://www.westwindcos.com/pdf/sunserver_11900.pdf
[28] Lyer S, Rowstron A, Druschel P (200) Squirrel: A Decentralized Peer-to-Peer Web Cache. In: Proc. 21th ACM Symp. on Principles of Distributed Comp., 213–222
[29] Moore D (2001) The Spread of the Code-Red Worm (CRv2). http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml
[30] Nah F (2004) A Study on Tolerable Waiting Time: How Long Are Web Users Willing to Wait? In: Behaviour and Information Technology, 23(3):153–163
[31] Niven L (1973) Flash Crowd. In: The Flight of the Horse, Ballantine Books, 99–164
[32] Padmanabhan VN, Sripanidkulchai K. (2002) The Case for Cooperative Networking. In: Proc. 1st Int. Workshop on Peer-to-Peer Systems, 178–190
[33] Pan C, Atajanov M, Hossain MB, Shimokawa T, Yoshida N (2006) FCAN: Flash Crowds Alleviation Network Using Adaptive P2P Overlay of Cache Proxies. In: IEICE Trans. on Communications, E89-B(4):1119–1126

[34] Pan C (2006) Studies on Adaptive Network for Flash Crowds Alleviation. Ph. D. Thesis, Saitama University
[35] Park K, Lee H (2001) On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. In: Proc. ACM SIGCOMM 2001, 15–26
[36] Ratnasamy S, Francis P, Handley M, Karp R, Shenker S (2001) A Scalable Content-Addressable Network. In: Proc. ACM SIGCOMM 2001, 161–172
[37] The Ring Server Project. http://ring.aist.go.jp/index.html.en
[38] Rowstron A, Druschel P (2001) Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility. In: Proc. ACM 18th Symp. on Operating Systems Principles, 188–201
[39] Rubenstein D, Sahu S (2001) An Analysis of a Simple P2P Protocol for Flash Crowd Document Retrieval. Tech.Report, EE011109-1, Columbia Univ.
[40] Saroiu S (2001) Bottleneck Bandwidths. http://www.cs.washington.edu/homes/tzoompy/sprobe/webb.htm
[41] Shimokawa T, Yoshida N, Ushijima K (2000) Flexible Server Selection Using DNS. In: Proc. Int.Workshop on Internet 2000, in conjunction with IEEE-CS 20th Int. Conf. on Distributed Computing Systems, A76–A81
[42] Shimokawa T, Yoshida N, Ushijima K (2006) Server Selection Mechanism with Pluggable Selection Policies. In: Electronics and Communications in Japan, III, 89(8):53–61
[43] Sivasubramanian S, Szymaniak M, Pierre G, Steen M (2004) Replication for Web Hosting Systems. In: ACM Comp. Surveys, 36(3):291–334
[44] Stading T, Maniatis P, Baker M (2002) Peer-to-peer Caching Schemes to Address Flash Crowds. In: Proc. 1st Int. Workshop on Peer-to-Peer Systems, 203–213
[45] Stavrou A, Rubenstein D, Sahu S (2004) A Lightweight, Robust P2P System to Handle Flash Crowds, In: IEEE J. on Selected Areas in Comm., 22(1):6–17
[46] Stoica I, Morris R, Karger D, Kaashoek F, Balakrishnan H (2001) Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proc. ACM SIGCOMM 2001, 149–160
[47] Wang J (1999) A Survey of Web Caching Schemes for the Internet. In: ACM Comp. Comm. Review, 29(5):36–46
[48] Wang L, Pai V, Peterson L (2002) The Effectiveness of Request Redirection on CDN Robustness. In: ACM Operating Systems Review, 36(SI):345–360
[49] Welsh M, Culler D (2003) Adaptive Overload Control for Busy Internet Servers. In: Proc. USENIX Conf. on Internet Technologies and Systems
[50] Zao W, Schulzrinne H (2004) DotSlash: A Self-Configuring and Scalable Rescue System for Handling Web Hotspots Effectively. In: Proc. Int. Workshop on Web Caching and Content Distribution, 1–18
[51] Zona Research, Inc.(1999) The Economic Impacts of Unacceptable Web-Site Download Speeds. White Paper, http://www.webperf.net/info/wp_downloadspeed.pdf

第12章　基于CDN的协作流媒体服务

Giancarlo Fortino, Carlo Mastroianni 和 Wilma Russo

12.1　引言

近年来，内容分发网（CDN）已经成为了一种互联网流媒体服务的高效解决方案，从TV广播到视频点播其应用遍布互联网[1]。然而，现代多媒体应用并不仅仅是提供内容的检索或访问，还包括内容的创建、修改和管理，并能够主动地寻找合适地点来放置内容，借此提供新的增值服务[2]。

CDN可以有效地用于支持协作流媒体服务，特别适用于协作播放服务[4]，该服务能够使一组客户端以某种共享的方式来请求、观看流媒体和控制流媒体的会话。

本章介绍了一种基于CDN的网络架构，能够支持协作式的播放服务。从流媒体的分发和控制而言，该服务能够显著提升那些支持协作播放服务的中心化架构的性能。具体而言，本章提出了一种称为分层式合作控制协议（HCOCOP）的服务架构，能够控制在CDN支持下的协作播放服务会话中共享的流媒体。在对协作播放会话进行设置时，HCOCOP被映射到分层控制结构中，该结构由基于CDN的架构组成并提供支持，分层结构由一台位于底层的协作服务器、一台或多台位于中间层的控制服务器以及位于叶节点层的客户端组成。

HCOCOP通过离散时间仿真来进行实现和性能评估。具体而言，性能评估环节（包括对称和非对称的控制结构拓扑）以及三种不同的HCOCOP架构（NoCoop、LocalCoop和GlobalCoop）能够针对阻止概率和拒绝概率这两个性能指标进行分析，而这两个指标具体表征了HCOCOP的性能。

本章余下内容组织如下：首先描述提供协作播放服务的架构；然后介绍一种理论上的CDN，称为COMODIN；12.4节和12.5节分别给出对HCOCOP的描述和性能评估；在12.6节，描述了基于CDN协作播放服务系统的两个应用领域；12.7节给出未来的研究方向；在最后一节中，对本章要点进行总结。

12.2　背景知识和相关工作

协作播放服务使得明确组成的同步用户组能够选择、观看并协作地控制一个远程的媒体播放。通过该服务支持的会话被称为协作播放会话（CPS）[4]。

特别而言，一个CPS包括三个紧密相关的会话：

(1) 多媒体会话

一个以录制的音频/视频报告文件（如一次讨论组会），或一部电影，或者某个

更复杂的合成多媒体对象的形式呈现的媒体播放功能被同步传递给组内所有成员，允许所有成员观看。

（2）控制会话

媒体播放由 VCR 的典型指令控制（如播放、暂停、拖动等），因此组内任意成员都能够改变多媒体会话的状态。

（3）交互会话

组成员可以彼此交换有关组建和共享媒体播放的相关知识。

一个支持协作播放服务的架构需要以下核心服务来组织和运行 CPS：

（1）组的形成和管理（GFM）

GFM 服务支持协作组的形成和管理。具体而言，一个组将围绕着由 CPS 组管理员和受邀用户的注册所决定的媒体对象的选择结果来形成。

（2）流媒体（MS）

流媒体服务支持从一个选定的媒体文件对象到所有组成员的基于流的分发。

（3）控制流（SC）

控制流服务允许组成员来控制那些流媒体支持的多媒体会话。

（4）组交互（GI）

组交互服务通过基于文本的报文方式支持组成员之间的信息交换。

具体而言，SC 服务基于播放控制协议，该协议允许通过发送类似于 VCR 的控制指令，并负责在控制指令影响 CPS 时处理 CPS 状态的改变。为了监督组成员在发送控制指令时的行为，流控制协议必须使用合作机制。这些合作机制能够决定组中哪个成员有权发送控制指令（在基于底层的合作时）或哪一条传输控制指令被接受（在基于随机的合作机制时[3]）。基于底层的合作依赖于令牌环的概念，该令牌环必须被一个组成员明确获取，用于发送控制指令。相反，在基于随机的合作方式中，每一个组成员都可以在无需请求底层的情况下发送控制指令，因此被不同组成员同时发送的控制指令之间会出现冲突。所以，系统需要决定应该接受哪个指令。

另外，对 CPS 状态变化的控制也是一个关键操作，因为它涉及对播放状态的修改和将此修改信息向所有的组用户持续传输。

为了描述 CPS 控制处理机制和状态改变处理机制，我们使用了图 12.1 中的简明星形架构作为参考。图中组件分别为：①媒体流和控制服务器（MCS），负责集成流媒体服务 MS 和流控制服务 SC；②组播通信信道（MCC），支持流媒体和控制指令的传输；③协作客户端（CC），负责同一组内成员之间的协作。CPS 的播放状态由 MCS 管理，当收到控制指令后 CPS 状态发生改变。播放状态的自动机如图 12.2 所示，包括两个状态（播放和暂停）和相关控制指令标记出来的状态转移（播放、拖动、暂停、停止）。协作客户端也包含一个相同形式的自动机镜像，在 MCS 的自动机状态发生改变时镜像也同步进行改变。因此，当 MCS 改变了播放状态自动机时，它将发送一条更新报文给所有客户端，从而使得它们能够相应改变自身的自动机。在此更新过程中，MCS 不考虑其他的控制指令。

为了举例说明 MCS 和 CC 之间的交互关系，我们使用图 12.3 中所示的时序图，

在图中假定协作机制是随机的。我们主要考虑三种情形：

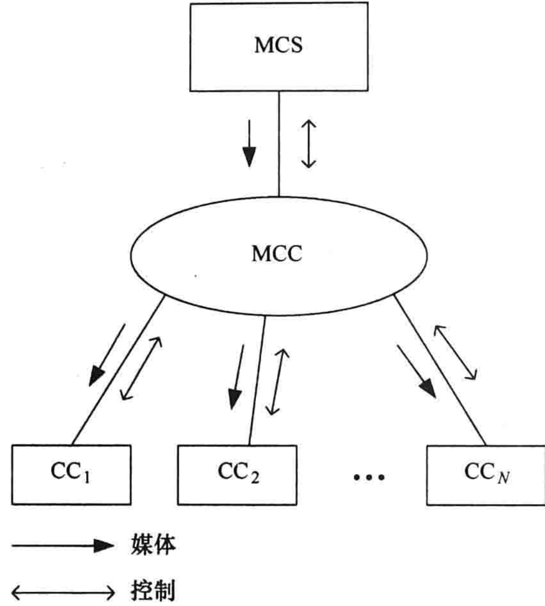


图 12.1 作为参考的星形简要架构

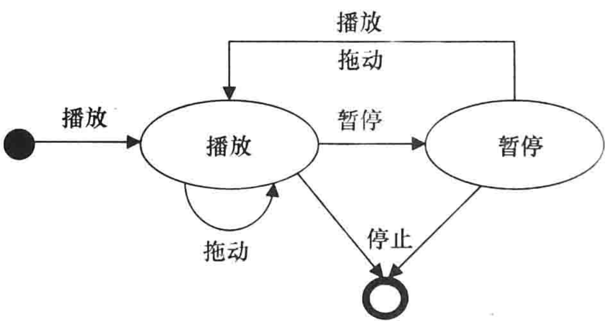


图 12.2 CPS 播放状态的自动机

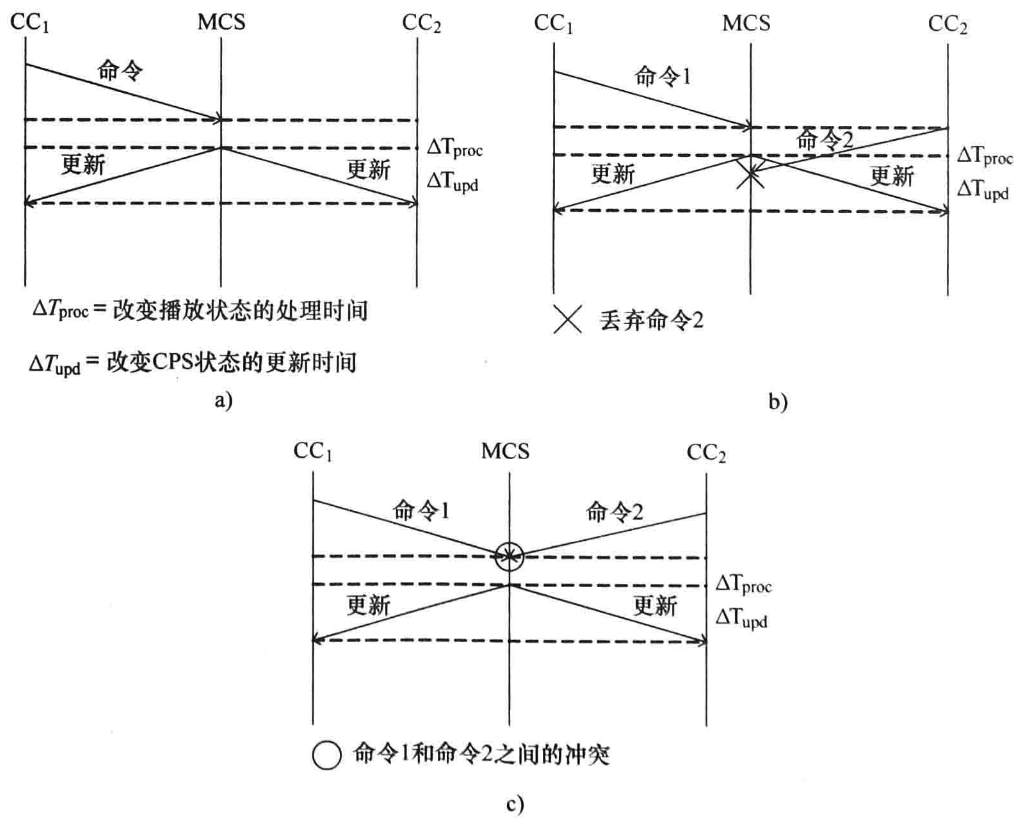


图 12.3 MCS 和两台 CC 之间交互的时序图
a）没有冲突 b）没有冲突及丢弃指令 c）有冲突

1）无冲突情况，其中 CC_1 发送控制指令。

2）无冲突情况并且指令丢弃，其中 CC_1 发送指令 1，CC_2 在指令 1 发送后再发送指令 2。

3）有冲突情况，CC_1 和 CC_2 都同时地发送控制指令。

在第一种情况中，CC_1 发送一条控制指令，MCS 在收到指令后对其进行处理，改变播放状态，并将更新信息传递给所有的协作客户端。

在第二种情况中，MCS 在控制指令处理和 CPS 状态更新两个过程中拒绝任何其他控制指令。

在第三种情况中，CC_1 和 CC_2 发送的控制指令在同一时间抵达 MCS，因此 MCS 必须决定哪一条控制指令需要接受，并同时丢弃另外一条。之后，MCS 按第一种情况进行操作。

目前，已出现一些系统设计和实现 CPS 的功能，它们的架构可以分为基于星形的结构（如上文所述）和基于 CDN 的结构。

MBone VCR 点播[9]、MASH Rover[13] 和 ViCROC[4] 等系统依赖于星形架构，其中一台中心服务器负责整个组的组织、基于 IP 组播的流媒体传输和流控制。具体而言，流媒体传输基于 IP 组播并面向所有系统。流媒体控制使用 IP 单播的仅为 MBone VCR 点播系统，而其他两个系统用的则是 IP 组播。集成在这些系统上的流控制协议使用随机协作机制，通过接受首条输入控制指令并丢弃其余控制指令的方式来解决冲突问题。以上提到的系统会遇到两个主要问题：中心服务器的性能瓶颈，以及 IP 组播稀疏可用性导致的难以部署的问题。

另外，最近出现了一种新的基于星形架构的流控制协议，称为协作控制协议（COCOP）[6]。COCOP 依赖于与上述系统相似的随机机制，但是它也使用了另一种协作机制，即一个组成员在得知其他成员已发送控制指令后，会避免再向系统发送任何控制指令。Fortino 等人[6] 已经证明，使用这种协作方法能够提升系统的整体性能。

COMODIN 系统[7] 提供了与星形系统相同的功能，但是它是基于 CDN 架构的，该方法不仅克服了上述系统的问题，并且同原有系统相比，能够增加流媒体控制的效率。

12.3　COMODIN 系统概览

本节将对 COMODIN 系统进行简要介绍。COMODIN 系统的架构由两个平面组成（见图 12.4），一个是包含提供媒体点播流服务的流 CDN（SCDN）的基平面，另一个是提供协作播放服务的协作平面。

基平面包括以下基本网络组件：

1）源服务器，它存储了需要通过 CDN 发布的媒体对象。

2）代理缓存服务器（surrogate），它是源服务器内容的部分副本，具有短期地存储内容以及通过接入网将内容使用流媒体服务器组件分发给用户的能力。

3）客户端，它是一个向 CDN 请求特定内容的多媒体应用。

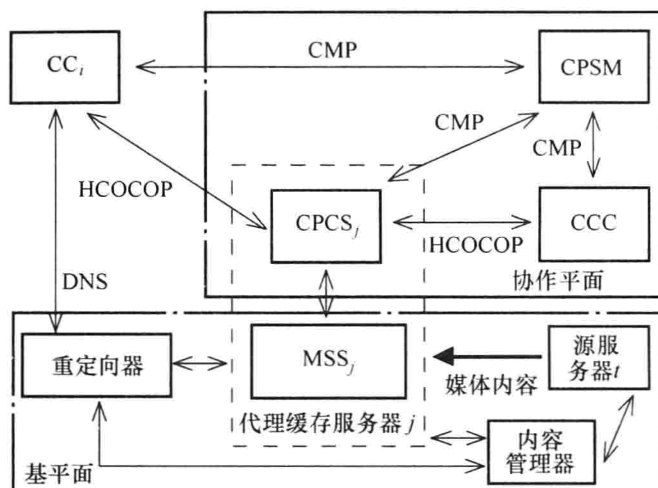


图 12.4 COMODIN 架构

4）重定向器，负责基于重定向算法为每一个不同的客户端请求选择最合适的代理缓存服务器[10]。

5）内容管理器，负责在代理缓存服务器和源服务器之间实现媒体内容的协作存储。

协作平面负责提供协作播放服务，由以下网络组件构成：

1）协作播放会话管理器（CPSM），基于协作播放会话管理协议（CMP）提供组的构建和管理核心服务。具体而言，CPSM 允许协作播放会话（CPS）的形成、注册/注销、初始化、加入/离开和结束等。

2）协作播放控制服务器（CPCS），同基平面上的 MSS 集成在一起，支持被一个 CPS 内的所有成员所共享的流媒体的远程控制。

3）CPCS 协作信道（CCC），能够通过协作信道协议（CCP）实现同一个 CPS 的所有分布式 CPCS 之间的协作。

4）协作客户端（CC），本质上是基平面上客户端组件的增强版，充当协作播放服务与用户之间的对接。

一个由 COMODIN 架构支持的 CPS 可通过以下步骤来建立并运行：

1）组织。一个组织者 CC 与 CPSM 相连，并请求建立一个 CPS。

2）邀请。组织者 CC 邀请其他 CC 加入，并通过直接报文来注册组织好的 CPS。

3）注册。受邀 CC 与 CPSM 建立连接，并在 CPS 上进行注册。

4）初始化。组织者 CC 与 CPSM 建立连接，请求对 CPS 的初始化，并将报文重定向到 CPCS。

5）加入。CC 通过注册的方式成为 CPS 的成员，报文被重定向给各自的 CPCS。

6）执行。任意成员发送播放控制请求来启动 CPS。一个 CPS 的状态通过一系列相继的控制请求（暂停、播放、拖动）改变。从控制角度而言，这一步是通过下一节所述的 HCOCOP 来操作。

7）终止。CPS 可通过投票机制被组织者 CC 终结。

图 12.5 给出了一个 CPS 的例子。该 CPS 由 COMODIN 系统和 4 个用户组成，每两个用户连接到一个不同的 CPCS（CPCS A 或 CPCS B），从而分别组成两个小组（A 和 B）。

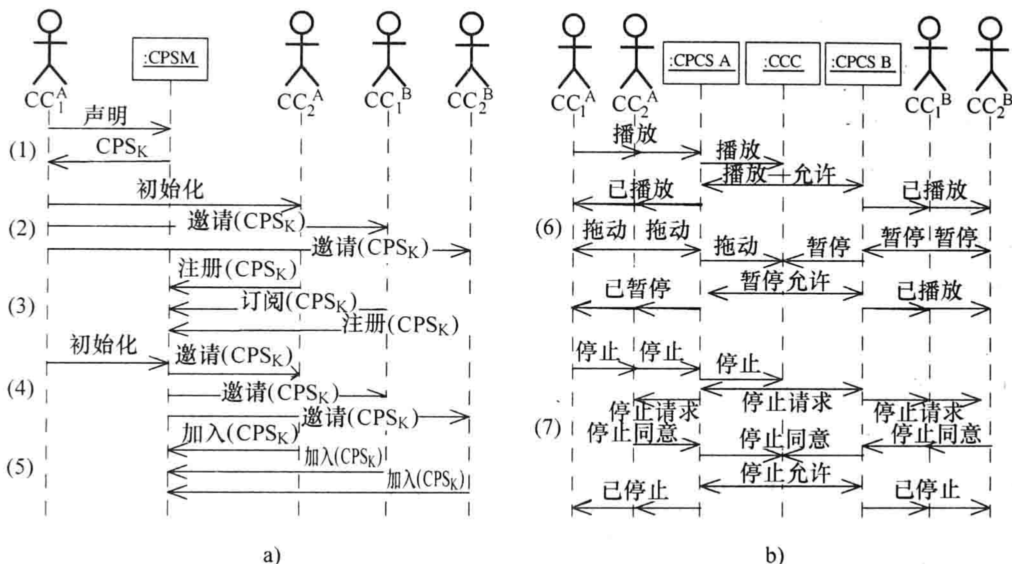


图 12.5　CPS 工作示例

a）CPS 建立　b）CPS 的运行

以下 1）~7）步分别给出了上述各步骤之间的交互（换言之，处于活动状态的组件之间报文序列的交换）：

1）隶属于小组 A 的某一客户端 CC_1^A 组织起一个 CPS（以下称为 CPS_K）。

2）CC_1^A 邀请其他三个客户端（CC_1^B、CC_2^A、CC_2^B）注册到 CPS_K。

3）CC_1^B、CC_2^A、CC_2^B 注册到 CPS_K。

4）CC_1^A 初始化 CPS_K。

5）CC_1^B、CC_2^A、CC_2^B 加入 CPS_K。

6）CC_1^A 开始媒体播放。在某一时刻，假设 CC_1^B 请求“暂停”，几乎与此同时 CC_2^A 请求“拖动”，那么 CC_1^B 将胜出，因为它的指令要先于其他指令抵达。

7）CC_1^A 发起一次投票来终止 CPS_K，CC_1^B、CC_2^A、CC_2^B 表示同意。

12.4　HCOCOP

HCOCOP 是 COCOP[6] 在 CDN 型架构上的拓展，它具备如下特性：

1）基于随机的控制指令传输机制。每一个组成员可在任意时间发出控制指令。组成员之间通过避免显式同步机制（如基于底层的协作机制）来增加系统的交互性，虽然这样可能会导致控制指令之间的冲突。

2）避免冲突的 FCFS 策略。如果两个或者多个控制指令被组内不同成员近似（准）同时发出，则通过 FCFS 策略选出一个胜出者来驱动 CPS 改变状态，其余指令

将被丢弃。

3）基于协作的降低无效控制指令传递率的机制。一个组成员在检测到其他组成员发出了控制指令时，它将避免再发出控制指令。这种机制降低了指令冲突的可能性。

4）基于软状态的 CPS 状态管理。一旦一个控制指令改变了播放状态，CPS 状态就会通过报文和计时器进行更新，并不需要硬状态的管理。

HCOCOP 被映射到 CPS（以下称为 CPS_K）的分层控制架构上，如图 12.6 所示。图中也给出了自动机（定义了协议行为）的位置。当 CPS 执行时，控制结构的组件由 COMODIN 协作平面上的架构控制组件派生：对于 CPS_K，CCC_K 是 CCC 组件的前端，$CPCS_i^K$ 是第 i 个 CPCS 的前端，$C_x^{K,i}$ 是 CPS_K 中第 x 个协作客户端，且由第 i 个 CPCS 前端提供服务。

HCOCOP 基本运行原理如下：如果客户端 $C_x^{K,i}$ 发送控制指令（ClReq），其对应的 $CPCS_i^K$ 在接受指令之前会将这个 ClReq 转给 CCC_K 来解决潜在冲突问题，即判断是否关联到其他 $CPCS_w^K$（$w \neq i$）的客户也几乎同时发送了一个 ClReq 指令。CCC_K 接收第一个输入的 ClReq，并通知所有 CPCS，并在一个给定的时间段内丢弃其他所有用户请求，这样就可以规范用户的交互性和避免会话死锁。与此 $CPCS_i^K$ 相关联的其他用户所引起的潜在冲突，则是由 $CPCS_i^K$ 采用与 CCC_K 相同的策略来解决。

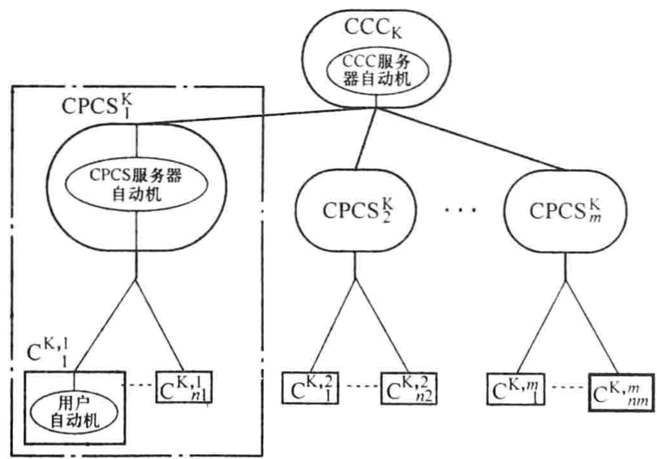


图 12.6　CPS_K 的基于 CDN 的控制架构

HCOCOP 可以运行在以下三种协作模式中：

1）全局协作（GlobalCoop）。ClReq 一方面通过 $CPCS_i^K$ 向下转发给所有与其相连的客户端，另一方面通过 CCC_K 转给所有的 $CPCS_w^K(w \neq i)$，并由后者转给所有相连客户端。此类机制允许一个用户检测到来自其他客户端的 ClReq，借此避免发出可能被系统丢弃的 ClReq 请求。

2）局部协作（LocalCoop）。ClReq 仅仅通过 $CPCS_i^K$ 向下转发给所有与其相连的客户端。

3）无协作（NoCoop）。ClReq 并不转发给任何其他用户。

定义 HCOCOP 行为的自动机如图 12.7 所示。当用户发出控制指令（UsrReq）时，客户端 $C_x^{K,i}$ 的自动机（见图 12.7a）发出一个客户端请求（ClReq），并进入就绪状态。在就绪状态的客户端 $C_x^{K,i}$ 感知到与同一个 $CPCS_i^K$ 相连的其他客户端也发送出 ClReq 请求时（局部协作模式）或者其他与 $CPCS_w^K(w \neq i)$ 相连的客户端也发出 ClReq 请求时（全局协作模式），系统也能进入请求完成状态。在该状态（自动机一直保持该状态，直到收到回复），系统对其他客户端发出的 ClReq 请求都不予受理，而用户 $C_x^{K,i}$ 被禁止发送新的控制请求，目的是降低会话负载。当回复抵达时，系统开始进行请求处理。为了控制会话的交互性程度，新用户的控制指令会在一个给定的生存周期 T_{CC} 内被阻止。

图 12.7 中，$CPCS_i^K$ 的 CPCS 自动机可以在就绪状态下接收 ClReq，ClReq 的接收意味着自动机进入了同步状态。如果 ClReq 来自于与它相连的客户端，那么该 ClReq（或者称为上行 ClReq）被转发给 CCC 服务器自动机。并且，如果局部或者全局协作模式得到应用，它也会被转发到其他与之相连的客户端上去。如果 ClReq 来自于 CCC（例如，由连接到其他 CPCS 服务器上的客户端产生的 ClReq），这样的 ClReq 请求（或者称为下行 ClReq）被转发给所有相连客户端。在同步或者就绪状态下，一旦接收到来自 CCC 服务器自动机的回复，CPCS 自动机处理该回复并将其转发给所有相连客户端。随后，自动机便进入了处理完成状态，该状态在一个给定生存周期 T_{CPCS} 内会一直保持。引入该延时的目的是使得客户端意识到会话状态中的所有改变，进而利用类似于软状态的模式[12]来管理组的交互性。

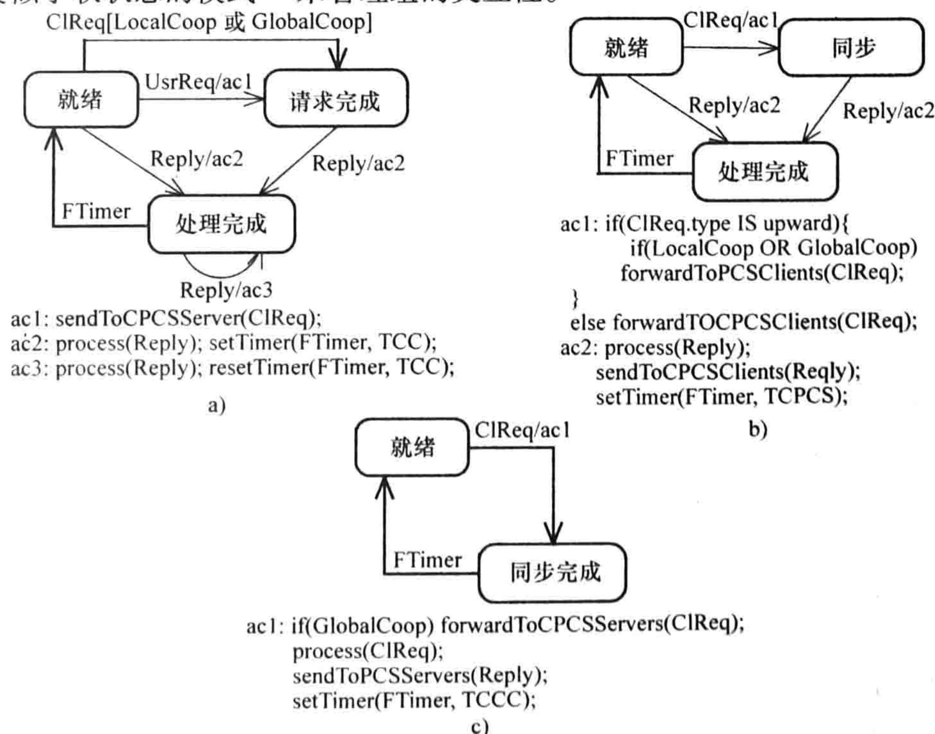


图 12.7　HCOCOP 的自动机

a）客户端自动机　b）CPCS 自动机　c）CCC 服务器自动机

CCC 服务器自动机（见图 12.7c）在收到由位于就绪状态下的 $CPCS_i^K$ 的 CPCS 自动机发出的 ClReq 请求之后，如果当前采用的是全局协作模式，那么它将接受请求并转发给所有其他的 CPCS 自动机。一个回复被送给所有 CPCS 自动机，且 CCC 服务器自动机变为同步完成状态，该状态在一个生存周期 T_{CCC} 内会一直保持。引入这个延时的目的在于确保 HCOCOP 的连续性。

12.5 HCOCOP 的仿真分析

本节对 HCOCOP 的性能进行分析，目的是证明在 CDN 中采用协作方法带来的性能优势。本节使用了面向对象的离散时间仿真框架[6]来对 HCOCOP 进行实现，并在 CDN 架构中通过不同数量的客户端和不同的拓扑条件对其性能进行评估。HCOCOP 性能也同时与星形架构中的非协作协议和协作协议进行了对比。结果显示，与星形架构相比，使用 CDN 的架构的确能够提高流控制效率。

12.5.1 性能指标

一个协作播放控制协议必须确保协作播放会话（CPS）的连续性，并且同时给予用户改变播放状态的能力。性能指标可以体现出一个协作播放控制协议的相关特性，应考虑到对用户发送控制指令的请求处理和对已发送的控制指令的处理（可参见 12.4 节）：

1）客户端自动机允许或者阻止用户请求，从而确定是否像 ClReq 那样被发送或阻止。在进程完成状态中用户请求被阻止，可确保 CPS 的连续性。而在请求完成状态，用户请求被阻止是因为需要分配用户请求优先级的协作机制正在起作用。

2）未被阻止的用户请求首先作为 ClReq 被转发给客户端自动机，随后再转发到相应的 CPCS 服务器，如果被接受，将继续被 CPCS 服务器转发到 CCC 服务器，在那里再决定是否接受。

在此基础上我们定义两个性能指标，即阻止概率（P_{BLK}）和拒绝概率（P_{DEN}）。根据上述第 1）点，阻止概率被定义为用户请求被客户端自动机阻止的概率；根据上述第 2）点，拒绝概率定义为 ClReq 被 CDN 丢弃或者拒绝的概率。特别地，定义不同的拒绝概率为：①P_{DEN}（CPCS），定义为 ClReq 被相应 CPCS 服务器丢弃的概率；②P_{DEN}（CCC），定义为 ClReq 被 CCC 服务器丢弃的概率；③P_{DEN}（CDN），有时也称为全拒绝概率，定义为一个客户端请求被 CPCS 服务器或 CDN 的 CCC 服务器丢弃的概率，计算公式为 P_{DEN}（CPCS） + （1 − P_{DEN}（CPCS））P_{DEN}（CCC）。

拒绝概率应当越小越好，因为服务器拒绝请求的现象在发出该请求的用户看来绝非好事。事实上，当一个用户请求被转发给网络时，尽管发出该请求的用户充分意识到他/她不可能控制服务器，但依然期望其请求能够被接受。阻止概率也应当被设置为一个足够低的值，因为它实际上表示了用户发送控制请求的无效性。然而，拒绝概率比阻止概率更重要，因为一般而言，相比无法发送控制指令，发出的控制请求被拒绝让用户更加无法忍受。

12.5.2　仿真参数

本节描述了仿真系统中的参数和相关设置，目的是定义一个相对真实的仿真环境，使得在 CDN 控制架构下对 HCOCOP 的评估尽量真实可靠（12.4 节中涉及部分内容）。

首先，考虑更一般化的参数，如每一次仿真会话的持续时间和用户活跃度；其次，重点关注运行能够定性描述 CDN 控制架构的参数（如链路延时、处理延时、定时器等）和星形架构的参数，供比较使用。

1. 一般参数

每一次仿真运行时间，即仿真会话期 $T_{SESSION}$，被设置为一段允许针对一个预定的统计相关度得出性能指标的时间（例如，至少 0.95 的概率说明统计误差在 5% 以内）。

同一个用户发出的两个连续请求的平均抵达间隔时间（用户活跃度）由平均请求抵达间隔时间（MRIT）来描述，MRIT 通过基于 Gamma 概率分布函数的统计模型进行建模[11]。特别地，用户活跃度被分为 5 个等级非常低（MRIT ≥ 15min）、低（10min ≤ MRIT < 15min）、中等（5min ≤ MRIT < 10min）、高（120s ≤ MRIT < 5min）、非常高（MRIT < 120s）。为了能够完全地对 HCOCOP 中从高到非常高的用户活跃度进行评估，MRIT 的值被设为在区间 {10s，180s} 中浮动。

2. CDN 参数

两个相邻节点之间的延时（δ）通过下列连接延时模型定义：

$$\delta_i = K_f\delta_m + N(K_v\delta_m,\ \sqrt{K_v\delta_m})$$
$$K_f + K_v = 1 \quad K_f, K_v \geq 0$$

式中，δ_m 是平均延时，而 δ_i 是一个给定报文的实时延时，由一个固定部分和一个变化部分构成。K_f 和 K_v 的值是固定部分和变化部分的权重系数，K_f 设置为 0.7。δ_i 的变化部分由服从正态分布的随机变量生成，其均值和方差设为 $K_v\delta_m$。该正态分布在 $-K_f\delta_m$ 处截断，以确保 δ_i 不为负。选择正态分布是根据 Gibbon 等人的相关研究[8]。延时模型的参数通过位于意大利和西班牙的 CDN 测试平台的测量值得到[7]。特别地，用户和相应 CPCS 服务器之间的连接延时 δ_m 设为 3ms，而对 CPCS 服务器和 CCC 服务器之间的连接设置为 61ms。为了公平比较，星形架构中用户和服务器之间的 δ_m 设为 64ms。

服务器处理延时（T_{PROC}）是 CDN 服务器（CPCS 或者 CCC）占用的时间或者星形服务器用来服务一个接收到的请求并且改变 CPS 相应状态的时间，T_{PROC} 被设置为 200ms。

服务器定时器（T_{CCC}、T_{CPCS}、T_{CC}）用于控制服务器的反应（T_{CCC} 和 T_{CPCS}）和系统整体的交互性。它们都设为 3.0s，与用户定时器 T_{CC} 相同。Fortino 等人[6]指出，这个设置值可以避免系统的死锁现象。

12.5.3 HCOCOP 的运行模式

本节中，分析 12.4 节中定义的无协作、局部协作和全局协作等运行模式，并对其进行比较。另外，也将上述结果与基于 COCOP 协议的星形架构的仿真结果进行对比。星形模式是目前中心化协作播放架构的代表，其中控制信息仅仅由单一服务器进行处理（见 12.2 节）。COCOP 协议有两种不同模式：协作（Coop）和无协作（No-Coop）模式，并通过两种自动机进行定义：COCOP 客户端进程自动机（与 HCOCOP 的客户端自动机很相似，参见 12.4 节）和 COCOP 服务器进程自动机，后者与 HCOCOP 的 CPCS 自动机很相似，但并不具备同步状态，因为在星形结构中没有必要和其他服务器进行同步。另外，星形结构系统中使用的控制协议（MASH Rover 和 ViCRO[C] 等，参见 12.3 节）都与 COCOP 处于无协作模式下的运行方式很相似，甚至可以被认为是这些协议的一个典型，因此也可以用于性能比较中。

12.5.4 性能评估

仿真阶段的目的是评估 HCOCOP 在一个简单的基于 CDN 的架构中的性能，该架构有两个小组和 12 个用户，这个规模对于协作播放会话而言已经足够大了，因为这种会话主要是针对中小型用户组设计[13]。

我们给出了使用对称拓扑的基于 CDN 架构的结果。然后，检测了在非对称拓扑和自适应情况下 HCOCOP 的性能，自适应情况是指一个用户从一个 CPCS 服务器动态重定向到另一个。

1. 对称拓扑结构的 CDN

本节进行了包括 12 个客户端和 2 个 CPCS 服务器的对称 CDN 的一组仿真。因为拓扑的对称性，每一个 CPCS 分配到 6 个客户端，如图 12.8 所示。

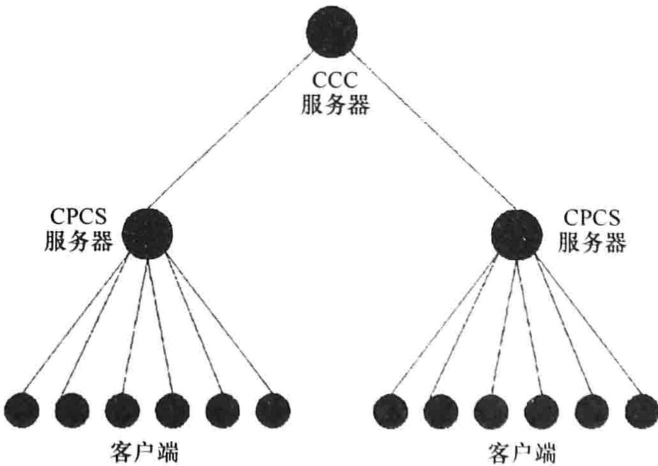


图 12.8 对称 CPCS 架构（分为两组，每组 6 个客户端）

图 12.9 显示了 CPCS 服务器上的拒绝概率 P_{DEN}（CPCS）。从图 12.9 中能够清楚地看出协作模式的优势。如 12.4 节中所述，当用户进程感知到连接到同一台 CPCS

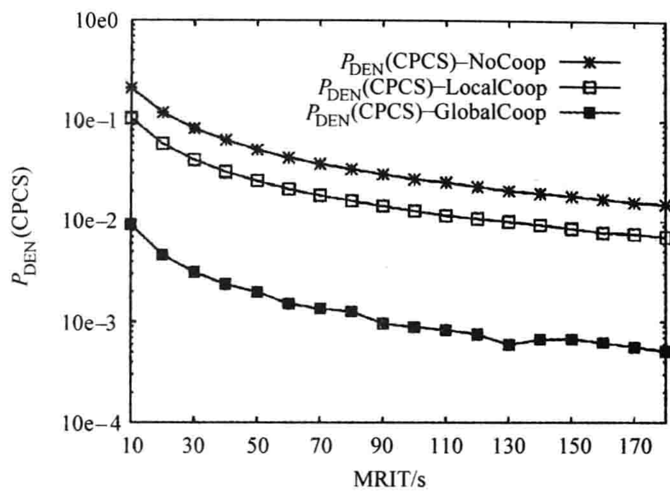


图 12.9　CPCS 服务器上的拒绝概率：无协作、局部协作和全局协作模式之间的比较结果

服务器的另一个客户端已发送请求时，局部协作模式便阻止该客户端发送控制指令。此模式的使用能极大地降低在无协作模式下的拒绝概率。在全局协作模式下，协作模式的好处得到了强化，因为连接到给定 CPCS 服务器的客户端也能够检测到连接到其他 CPCS 服务器的客户端所发出的请求。

图 12.10 表明 CCC 服务器中的拒绝概率 P_{DEN}（CCC）并没有被协作方法改善。不过，因为局部组级的优化，全拒绝概率的值 P_{DEN}（CDN）在全局协作模式下要相对低很多（见图 12.11）。CDN 和中心型架构（如星形）的拒绝概率也在图 12.11 中进行了比较。在 CDN 中，局部协作和非协作模式下的拒绝概率也与在星形架构中相应模式的拒绝概率不相上下。然而，在 CDN 全局协作模式下得到的拒绝概率要比其他情况下的概率低得多。因此，CDN 架构和协作机制相结合能够使客户端控制服务器的能力获得极大的提升。

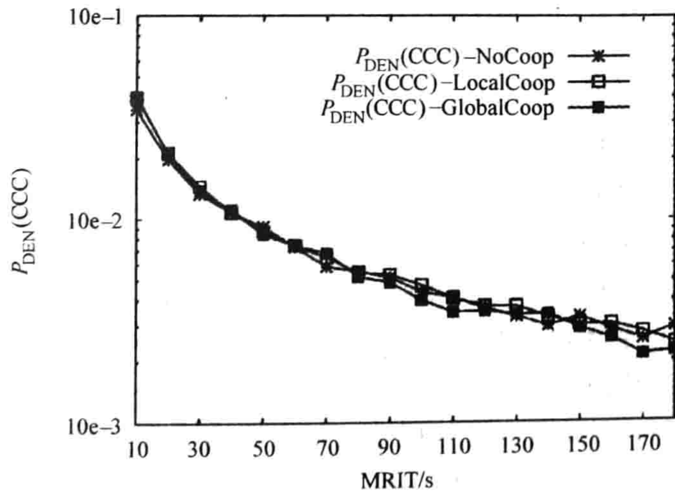


图 12.10　CCC 服务器中的拒绝概率：无协作、局部协作和全局协作模式之间的比较结果

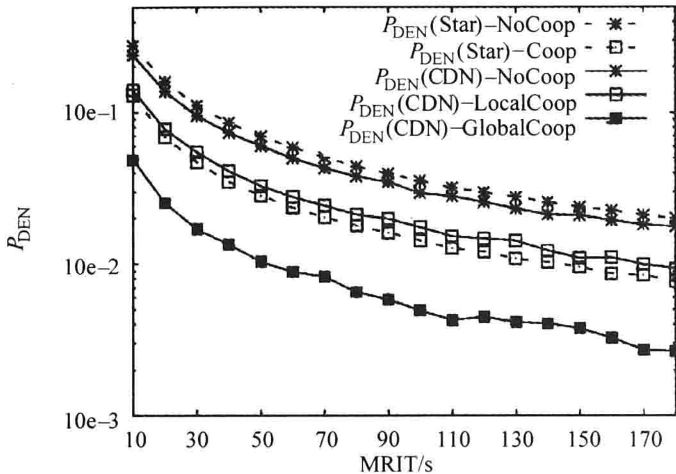


图 12.11　全拒绝概率：处于协作与无协作模式下的 CDN 和星形架构之间的比较结果

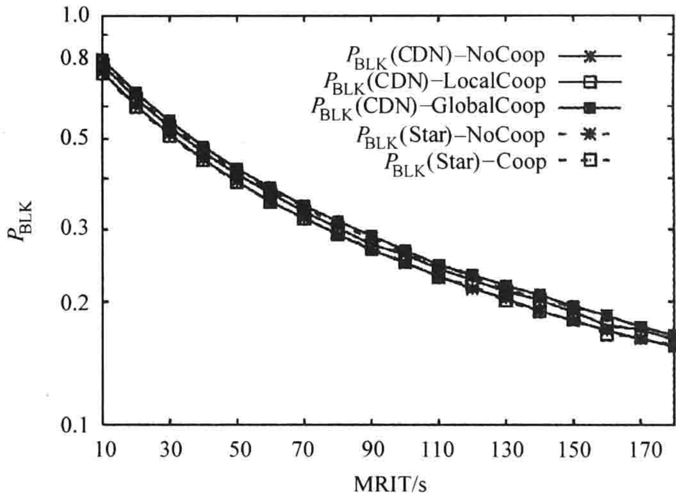


图 12.12　阻止概率：处于协作与无协作模式下的 CDN 和星形架构之间的比较结果

需要注意的是，虽然拒绝概率是主要性能指标，但该性能的提升是以牺牲阻止概率的代价获得的。图 12.12 表明，无论协作模式（无协作、局部或者全局协作模式）还是系统架构（星形或者 CDN），都未对阻止概率产生明显的影响。

2. 非对称 CDN 拓扑和动态客户端重定向

本节进行了一系列仿真实验，用来检测 HCOCOP 在一个由 12 个非对称分布于 2 台 CPCS 服务器之间的客户端所组成的 CDN 架构的性能，如图 12.13 所示。具体地说，7 个客户端分配给其中一台服务器，其他 5 个客户端被分配给另外一台。可以设想，在对称拓扑中一个客户端被一台服务器转移到（或者被重定向到）另外一台，就可以得到这种非对称拓扑结构。

为了更好地理解这种情况，可以回忆一下在 CDN 中如何使用一个请求路由算法将客户端的请求路由到一个合适的代理缓存服务器，在本例中也就是将客户端分配给特定的 CPCS 服务器。在自适应请求路由例子中[15]，可以根据 CDN 的当前情况动态地改变代理缓存服务器，这也是本例中将要考察的情况。下面将对此进行描述。

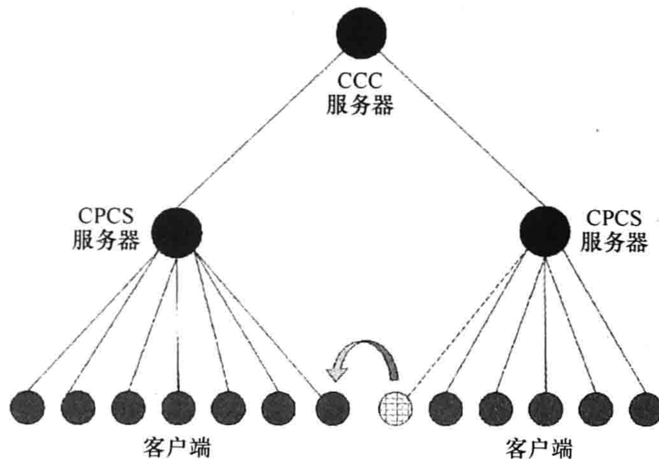


图 12.13　非对称性 CDN 架构（分为 2 组，每组有 7 和 5 个客户端）

图 12.14 给出了在协作模式和非协作模式下分属两个小组的客户端所感受到的全拒绝概率。比较结果显示，在无协作模式下两组的拒绝概率没有差别。然而，在协作模式下（局部协作和全局协作），隶属于较大组的客户端将会有更多的机会去控制会话状态。这个现象可以视为协作机制的优点。事实上，同一个小组内的客户端聚集在一起就可以提高小组所有参与者的性能。具体而言，该现象可解释如下：在局部层面，协作机制允许客户端感知到其他客户端发送的请求，因此，在同一个小组的客户端数量越多，客户端就会越能避免发送可能被本地 CPCS 服务器丢弃的请求，而这就能够均衡因局部客户端数增加而导致的局部并发性问题。另外，一旦属于较大组的一个局部获得了本地 CPCS 服务器的控制权，它就比其他组的客户端有更大的几率去控制 CCC 服务器。事实上，较大的组会转发给 CCC 服务器更多请求，因此，这些请求在 CDN 较高层面上遇到并发性问题的几率要比从较小组转发出的请求所对应的几率低得多。根据这个结果，客户端能够被有利地重定向到现有的组，而孤立的用户或者属于较小组的客户端会没有这个优势。

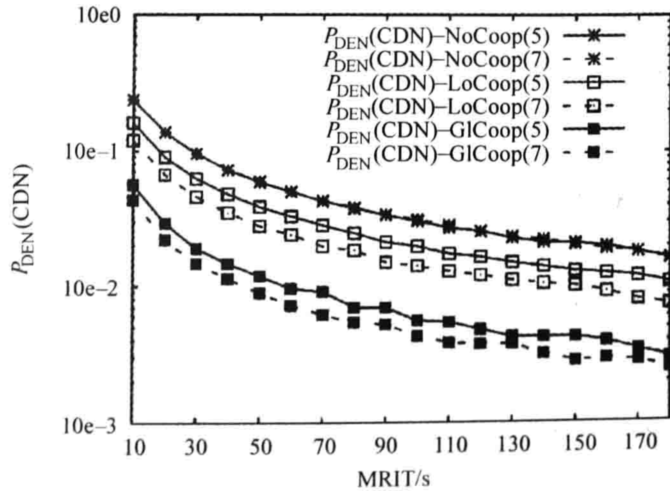


图 12.14　全拒绝概率：处于无协作、局部协作和全局协作模式下的非对称式 CDN 的比较结果

另外，两组客户端的阻止概率并没有表现出明显的区别。

图 12.15 显示了一个客户端从一个组被动态重定向到另一个组所产生的影响，此时，系统拓扑从每组原来 6 个客户端的对称结构变成了分别为 7 个和 5 个客户端的非对称结构。作为对图 12.14 中所给出结果的一种确认，当客户端被重定向到另外一个组后，该组的全拒绝概率降低了，同时原来所属组的全拒绝概率则有所增加。然而，对称拓扑情况下的拒绝概率处于中间位置。这也可以通过另外一种方式说明：如果初始配置是非对称的，那么客户端的重定向可以形成一个对称拓扑，这种方式能够在客户端之间实现更好的公平。

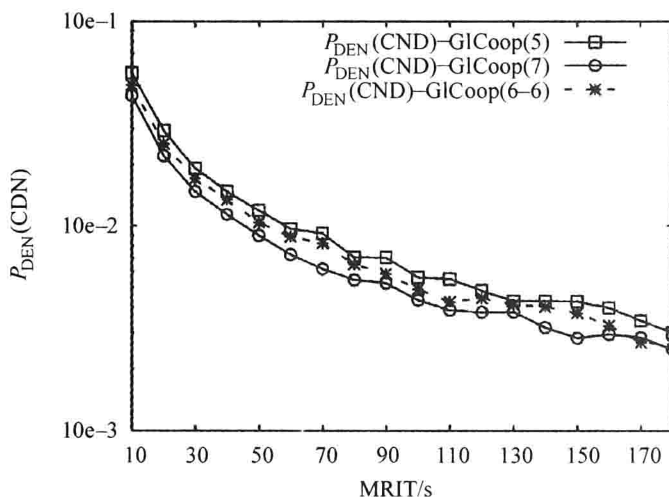


图 12.15 客户端重定向对全拒绝概率的影响：对称式 CDN 和非对称式 CDN 架构（由客户端从一台 CPCS 服务器重定向至另一台 CPCS 服务器导致）之间的比较结果

相对于拒绝概率，阻止概率几乎不会受到客户端重定向的影响。

从上面的结果可以看出，相对于拒绝概率，提高 CDN 架构的平衡性可以改进请求路由算法以及其他常用参数如网络邻近性、客户端—服务器延时、代理缓存服务器负载等。这些参数的组合最近成为了研究热点，其目的在于开发一个路由算法使其不仅能够提升数据的传输速率，更能够提升会话控制协议的效率。

12.6 写给使用者

基于 CDN 的架构所支持的 CPS 的开发和部署提供了在现有互联网设施上实现协作播放服务的可能性，也使得一些重要的应用领域从网络学习扩展到了网络娱乐。

基于 CDN 的 CPS 可以有效地支持协作式按需学习（CLoD）的网络学习项目[4]，这是一种虚拟的协作学习方法，能够令参与其中的学生发送请求、观看和控制课堂节目的播放和问题的交互，从而形成一个自主的交互式学习过程。CLoD 参考了教学视频辅导（TVI）和分布式教学视频辅导（DTVI）学习方法和工具的一些思想，在这些方法中一个学生小组在导师的指导下学习一段课程视频。DTVI 是 TVI 的完全虚拟

版本[14]，其中每个学生都有一台联网的计算机，配有视频（摄像头）和音频（麦克风和耳机）设备，用于在组内的彼此沟通。TVI 和 DTVI 的应用都已证明，参与实验的学生比真正课堂上的学生表现更好。CLoD 和 DTVI 之间的区别在于，CLoD 没有假设负责指导学生的导师的存在。事实上，在 DTVI 中只有导师能够控制视频会议记录设备（VCR），而在 CLoD 中每一个播放会话的参与者都能在分组的导师授课中使用共享的 VCR 遥控器。

基于 CDN 的 CPS 也能够应用于电子娱乐应用方面，如虚拟剧场，即在一个分布式的虚拟环境里用户的化身（虚拟自我形象）通过交换意见或者聊天来进行交互并协作观看或控制电影。

12.7　未来研究方向

本章所提到的基于 CDN 的架构，在提高服务效率和服务质量方面已进一步增强。特别地，本章所定义的 HCOCOP 在控制指令上没有区别，但是，将不同控制策略与不同控制指令相结合可以对协作播放会话实现更有效的控制。一个针对星形架构的多策略播放控制协议由 Fortino 等人[5]提出，其中作者定义了三种策略（分别基于随机、令牌和投票），并根据语义内容将其与相应的控制指令（暂停、播放、拖动和停止）联系起来。针对暂停控制指令的处理需要具备高度的交互性，因为它需要被 HCOCOP 的随机策略所支持。针对播放/拖动控制指令，则需要被令牌环机制支持，该机制允许持令牌者发送控制指令。最后，针对停止控制指令的处理（接受停止控制指令会终止 CPS）应当基于多数原则来完成，这就使投票机制得到有效的应用。

COMODIN 系统提供了在一个 CPS 的组成员之间的最优媒体流同步方法。CDN 或用户端的同步机制可以保证组内所有成员能够同步观看多媒体会话，但是这种同步机制尚未得到支持。目前的一个研究内容，是定义由 CDN 驱动的同步机制，该机制能够在不增加客户端负担的前提下提供多媒体播放的最优同步效果。

12.8　结论

本章提出了一种新型的 CDN 架构，它支持协作式流媒体服务，并允许显式形成的同步用户组来选择、观看和协作地控制一个多媒体会话。播放会话的控制由 HCOCOP 实现，且通过离散事件仿真对其性能进行了评估。仿真结果表明，与通常使用的星形架构相比，基于 CDN 的分层方法具有更高的效率，因为拒绝概率降低的同时，阻止概率并没有产生明显的影响。另外一个很有意思的结果是，在非对称拓扑中当客户端被分配到客户端数较多的组时，要比孤立客户端或者属于较小组的客户端得到更好的服务。如果与其他参数（如网络邻近性、客户端—服务器延时和代理缓存服务器负载等）相结合，那么就可以用于对 CDN 重要组件之一的请求路由算法进行优化。

参考文献

[1] Cranor, C. D., Green, M., Kalmanek, C., Shur, D., Sibal, S., Sreenan, C. J., Van der Merwe, J. E. (2001) Enhanced Streaming Services in a Content Distribution Network. *IEEE Internet Computing*, 5(4):66–75.

[2] Crowcroft, J., Handley, M., Wakeman, I. (1999) *Internetworking Multimedia*. Morgan Kaufmann Pub, San Francisco, USA.

[3] Dommel, H. P., Garcia-Luna-Aceves, J. J. (1999) Group Coordination Support for synchronous Internet Collaboration. *IEEE Internet Computing*, 3(2):74–80.

[4] Fortino, G., Nigro, L. (2003) Collaborative Learning on-Demand on the Internet MBone. In: Ghaoui C (ed) *Usability Evaluation of Online Learning Programs*. Idea Group Publishing, Hershey (PA), USA, pp 40–68.

[5] Fortino, G., Mastroianni, C., Russo, W. (2004) A Multi-Policy, Cooperative Playback Control Protocol. In *Proc. of the 3rd IEEE Int'l Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, USA, pp 297–302.

[6] Fortino, G., Mastroianni, C., Russo, W. (2005) Cooperative Control of Multicast-based Streaming On-Demand Systems. *Future Generation Computer Systems, The International Journal of Grid Computing: Theory, Methods and Applications* 21(5):823–839.

[7] Fortino, G., Russo, W., Mastroianni, C., Palau, C., Esteve, M. (2007) CDN-supported Collaborative Media Streaming Control. *IEEE Multimedia*, 14(2):60–71.

[8] Gibbon, J. F., Little, T. D. C. (1996) Use of Network Delay Estimation for Multimedia Data Retrieval. *IEEE Journal on Selected Areas in Communications*, 14(7):1376–1387.

[9] Holfelder, W. (1998) Interactive remote recording and playback of multicast videoconferences. *Computer Communications* 21(15):1285–1294.

[10] Molina, B., Palau C. E., Esteve, M., Alonso, I., Ruiz, V. (2006) On Content Delivery Network Implementation. *Computer Communications*, 29(12):2396–2412.

[11] Padhye, J., Kurose, J. (1999) Continuous Media Courseware Server: a Study of Client Interactions. *IEEE Internet Computing*, 3(2):65–72.

[12] Raman, S., McCanne, S. (1999) A model, analysis, and protocol framework for soft state-based communication. *ACM SIGCOMM Computer Communication Review*, 29(4):15–25.

[13] Schuett, A., Raman, S., Chawathe, Y., McCanne, S., Katz, R. (1998) A Soft State Protocol for Accessing Multimedia Archives. In *Proc. of the 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Cambridge, UK, pp. 29–39.

[14] Sipusic, M. J., Pannoni, R. L., Smith, R.B., Dutra, J., Gibbons, J. F., Sutherland, W.R. (1999) Virtual collaborative learning: a comparison between face-to-face Tutored Video Instruction (TVI) and Distributed Tutored Video Instruction (DTVI). (Technical Report N. SMLI TR-99-72 by Sun Microsystems Laboratories, Palo Alto, CA, USA).

[15] Wang, L., Pai, V., Petersen, L., (2002) The effectiveness of request redirection on CDN robustness. *ACM SIGOPS Operating Systems Review*, 36:345–360.

第13章 通过IP进行直播和点播视频服务的CDN

Miroslaw Czyrnek, Ewa Kuśmierek, Cezary Mazurek, Maciej Stroiński 和 Jan Weglarz

13.1 引言

现阶段通过宽带 IP 网络的视频点播服务和直播电视节目播放服务已经开始变得普及。用户希望在任意时间和任意地点通过身边现有设备以互动的方式访问高质量视频，而传统的电视分发平台难以满足这样的需求。另外，宽带 IP 网络中的内容分发技术能够让内容提供商提供增值服务，使真正交互式的内容访问成为可能。然而，设计大规模、基于 IP 的多媒体内容分发系统面临着挑战，主要来自数据量累积和总传输速率的限制。信号处理技术的进步可以在不增加带宽要求的前提下将高质量的信号传递给端用户，1Mbit/s 的速率被认为可以获得足够理想的传输质量。然而，在这样的速率下对大量用户提供实时服务，会带来另外的挑战。

内容分发网络（CDN）就是一种互联网应用的解决方案。在宽带 IP 网络上 CDN 的优势是显而易见的，但与针对传统 Web 内容的 CDN 和专门为多媒体内容所设计的 CDN 相比则存在一些区别。针对传统 Web 内容所设计的 CDN 一般只需要负责低质量内容流的直接分发服务。而另一方面，多媒体 CDN 的目的在于分发能够与传统广播媒体质量媲美的多媒体内容，并支持相关服务[6,7]。以上对于数字视频和音频传送的特殊性能的种种需求考虑都源于多媒体内容的特殊性。

本章讨论基于 iTVP 平台的多媒体 CDN 设计的关键问题。iTVP 是一个基于 IP 网的多媒体内容分发平台，能够面向全国对大量同时在线用户进行服务。iTVP 平台服务包括访问直播电视节目、视频点播和音频点播、时间平移、电子节目向导（EPG）、个人录像机（PVR）等。考虑到操作的范围和提供内容的自身特性，高效的内容分发对于平台的正常运转而言非常必要。因此，CDN 成为 iTVP 平台的核心组件之一。

我们设计的 CDN 中，关键在于系统的分层本质，其中副本服务器被置于高一级的层次中，而底层节点被放置在运营网络的“最后一公里”层（即与用户相连部分）。从整体上 CDN 被分为自治但相互协作的不同区域。我们将内容缓存和内容复制集成起来，并利用了处于 CDN 高层的协作复制的优势。内容分发以文件下载的模式执行并传至在低层节点处，再以流模式从低层节点传到端用户处。这种方法使得我们能够采用相对简单易行的流量模型，与使用 VBR 编码的视频模型相比更加方便，因此简化了带宽的管理。我们在每个区域内都使用中心内容目录来解决内容定位的问题。CDN 的监控子系统提供必要信息供系统选择最适合的节点，将其分配给指定用户以提供服务。

本章提出了 iTVP 的内容分发架构、功能性结构和运行原理，解释了副本服务器

的放置、内容分配和分发、用户请求路由的相关规则。CDN 的特点在于确保有效的资源利用、可靠性与可扩展性的机制，因此本文解决对资源（如带宽和存储）的需求问题，特别讨论了高质量多媒体流，并探讨了端用户处的服务质量（QoS）问题。

本章内容如下：在 13.2 节，介绍了背景知识和多媒体 CDN 的相关内容；对于 iTVP 的介绍和整体平台的概况性描述在 13.3 节中给出；接下来 13.4 ~ 13.6 节重点关注 CDN 功能上的一些关键问题，如副本服务器放置问题、内容分配与分发问题以及用户请求路由问题；系统中收集到的运行数据和相关性能分析在第 13.7 节给出；在第 13.8 节中给出未来的研究方向；13.9 节是对未来多媒体 CDN 的一些见解；最后，在第 13.10 节中总结整章内容。

13.2 背景和相关研究工作

面向多媒体内容的 CDN 和传统面向 Web 对象内容分发的 CDN 不同。多媒体内容具备和其他类型内容完全不同的特性，并且多媒体内容的本质特性会影响到 CDN 诸多方面的设计，如 CDN 拓扑架构、副本服务器的数量和放置地点、内容分配和分发等。本节对多媒体 CDN 在设计和运行时需要重点考虑的几个方面进行简要介绍。在 CDN 的设计和运行中需要考虑到多媒体的若干特性，这些特性可以大致分为两类：第一类是多媒体内容对象的特性，如典型数据容量和编码技术；另一类是分发和内容访问模式。

13.2.1 多媒体特性影响

从多媒体特性对 CDN 总成本的影响开始介绍，具体而言主要关注对与带宽相关的分发成本和存储成本的影响。多媒体对象的尺寸一般而言要远远大于 Web 对象。视频的大小取决于一系列参数，如分辨率和编码类型等，通常 1h 长度的视频其大小在数百 MB 或者几个 GB。另外，现在并没有一种通用视频编码速率能适用于所有用户。编码率越高，视频质量越高，因此应在可用带宽的约束下最大化编码率。视频一般会被编码成多种格式，使不同用户可以根据自己的网络带宽情况选择不同的质量格式。这样的方式会进一步增加视频对象大小。因此，副本服务器无法存储用户要访问的所有内容。用户可能需要通过不同的副本服务器来获得不同的对象，而非仅仅在一台最近的服务器上获取所有内容。存储空间必须被认为是一种有限资源，而且对于整体成本而言，当考虑服务器数量和放置问题时，存储问题变得尤为重要。不仅如此，大数据对象对于带宽的使用会带来更高的分发成本。因此，CDN 网内的分发成本，例如从源服务器到副本服务器以及从一个 CDN 节点到另一个节点的成本等，都应当作为从 CDN 节点直接到端用户的成本之外的额外的内容分发成本而加以考虑，即整个分发过程分为网内和网外两步。Yang 等人[22]分析了在多媒体 CDN 中，副本服务器数量对这种两步分发方法的成本的影响。他们的结论是，过多数量的副本服务器会导致 CDN 内的分发成本实际上被少数用户所承担，这样一来过多的服务器反而导致整体成本超出最优。因此，需要谨慎选择副本服务器的数量。

在 CDN 中，多媒体内容的另外一个至关重要的特性在于视频编码策略。多媒体内容不像 Web 页面内容那样频繁变化，因此缓存一致性的问题并不突出。然而，因为多媒体内容分发有确定的资源需求，大多数都与内容流分发的可用带宽有关，因此同一个多媒体内容可以在 CDN 内部存在多个不同质量的版本。特别地，内容可以被编码为分层形式，这样在带宽允许的范围内，用户具有能够接收尽可能多的层和尽可能高质量的选择。层可以被编码为一种视频质量逐步增加的分级递进形式，但必须逐级相结合，才能提升视频质量。因此，层与层之间并非分离的对象，而是紧密联系。对于每级的复制都需要做出相应的决策，Su 等人[20]提出了一种分级多媒体内容的复制方法，其目的在于减少内容访问时间和存储成本。

13.2.2　多媒体分发和访问模式影响

多媒体内容一般以流的模式进行分发，这使得用户能够在收到内容后的很短时间内立刻播放，这一点与内容下载模式完全不同。使用流作为分发模式对于 CDN 的运转而言意义重大，已经有很多相关技术用于提升内容流系统的可扩展性[1,10,13,14,15,19]。这些技术一般而言是基于组播的传输模式，优势在于当一个流被分发到大量用户时，能够降低服务器带宽的利用率。在网络层或者应用层中使用组播技术改变了以往单播环境下分发成本依赖于副本服务器数量的状况。在单播环境中，服务器总带宽并不依赖于副本服务器的数量或者它们的放置地点，仅仅取决于同时在线的用户数量，因为一个用户对应一个流。在组播环境中，服务器总带宽使用率随着副本服务器的数量增加而增加，因为用户请求被发往大量服务器，并且一个流通常定向给更少的用户。

网络总带宽和服务器总带宽不同，因为前者不仅仅考虑总传输速率还要考虑内容传递的距离。因此，在单播环境中，使用最短路径路由算法后，网络总带宽随着副本服务器数量增加而下降。在组播环境中，网络总带宽也依赖于接收每一个流的用户数量。因此，即便一条路径很长，如果大量用户共享，那么网络总带宽使用率依然会降低。

Alemida 等人[2,3]研究了内容分发总成本的最小化问题，包括网络总带宽和整体服务器带宽，并假设使用组播流技术实现从副本服务器到用户的内容分发。他们的研究表明，传统单播内容分发系统的成本要远高于最优值。他们考虑了如何确定副本服务器的最优数量、副本服务器的放置和请求路由的问题，解决这些问题可以使得在使用组播的系统中整体成本接近最优。Kusmierek 等人[18]研究了副本服务器数量和副本服务器放置地点对于整个网络的影响，以及在借助代理服务器的周期性广播技术的系统中服务器带宽使用率的问题。他们的结论表明，在只使用一台副本服务器时，服务器带宽需求可达到最小，并随着副本服务器数量的增加而变大。另外，网络带宽需求随着副本服务器数量的增加而下降。

可扩展流技术的应用不仅仅影响了需要确定的副本服务器数量，而且还与用于最小化系统开销的内容分配问题紧密相关。Griwodz[11]研究了一种分层 CDN 中的内容分配问题，其中系统使用组播和部分分发，并对视频片段使用无序分发技术。作者得出的结论是，基于最优化成本的副本服务器放置策略，即高效地使用组播的策略有可能

将热门的视频内容放置在远离端用户的位置、将访问量低的内容放置于端用户的附近，从而影响提供给用户的服务质量。这种对服务质量的负面影响可以通过修改成本计算和选择合适的数据流融合机制参数来消除。

多媒体 CDN 中的资源使用量在不同的时间尺度上变化很大。具体而言，CDN 上的电视广播在每日的不同时段呈现不同的变化，因为一般而言，用户在晚间看电视要比早上多得多。给定多媒体对象的典型大小，相对于 Web 对象，并发用户数的变化所导致的上述变化的幅度要高得多。Cahill 等人[5,6]研究了这样一个系统中动态变化的资源需求，他们假设 CDN 运行在一个共享网络而不是专用的私有网络。因此，用于内容分发的资源可以通过从服务提供商租赁得到，并在需要时可以释放。作者借此提出了一个 CDN 模型，该模型并不包括启动成本，但是基于一个假设，即在此模型中用于内容分发的副本服务器的数量可以动态改变，并使用了动态改变的存储空间和带宽需求。

最后，多媒体内容经常通过非序列方式进行访问，就像用户进行 VCR 操作那样。此类型的内容访问会影响到内容分配。如果无法在每台副本服务器上都复制整个内容库（由于存储空间的限制），那么仅仅存储某些和端用户相关的特定的视频片段会有助于内容分发。已有结果表明，此方法能够降低访问视频播放目录的延时，并减轻网络状态对内容播放的影响。使用 VCR 之类的操作可能需要对某些视频片段进行复制[12]，从而允许用户从一个视频片段跳至另外一个片段，而无须长时间的等待。此方法能够改善内容分发和用户请求的路由规则。

13.3 iTVP 平台

本节通过描述 iTVP 平台，给出了多媒体 CDN 设计的方法。在开始讨论 iTVP 平台的 CDN 相关特性之前，首先给出 iTVP 平台的功能性和组织性的描述。

iTVP 是一个多媒体分发平台，设计目的在于为任意互联网用户提供全国范围内的内容服务。该系统提供如下相关服务：实况 TV 节目传输、实况节目传输的时间平移（即时间条拖动）、视频点播、音频点播以及某些附加服务，如电视节目向导和个人录像机等。iTVP 平台从多个独立内容提供商处分发内容，并与最终网络运营商进行协同。内容以多种编码形式分发并通过多种端设备包括 PC、机顶盒和手持设备等获取。

内容分发系统是平台的核心组件之一。除此之外，平台还包括内容提供商系统、许可证管理系统和一些交互端口用做端用户的主访问点（见图 13.1）。内容提供商系统在 CDN 中对应的是源服务器，它不仅负责存储提供商的所有内容，还负责内容编码、元数据描述、确认、颁发许可证和内容分发。许可证管理系统的设计初衷是为了支持多许可证的情形，以及与交互端口协同的多种付费方式。

从功能上说，iTVP CDN 由以下子系统组成：分配、监控、管理和报告子系统。分配子系统负责多媒体内容的分配和分发，它执行内容缓存和实时流中继，并负责资源管理和访问控制。为了高效地工作，分配子系统使用由监控子系统在线收集并处理

所得到的系统节点参数。监控子系统主要负责实时传递涉及系统运行状态的各方面最新信息，包括硬件设备性能数据、网络接口参数、应用层负载、服务性能数据和可用节点情况。收集这些参数使得监控子系统能够在全局上进行系统的负载均衡、故障检测和系统性能评估。管理子系统的任务是存储和收集 CDN 节点配置、内部服务接入点、网络层的配置和由其他平台组件（如内容提供商系统）所提供的外部服务等相关信息。报告子系统收集由监控子系统采集并获取的实时监控参数，并存储系统运行期间发生的所有事件，供系统性能分析使用。iTVP CDN 的所有子系统是分布式的，并被部署于所有系统的各个节点处，彼此之间进行协同来提供系统服务。

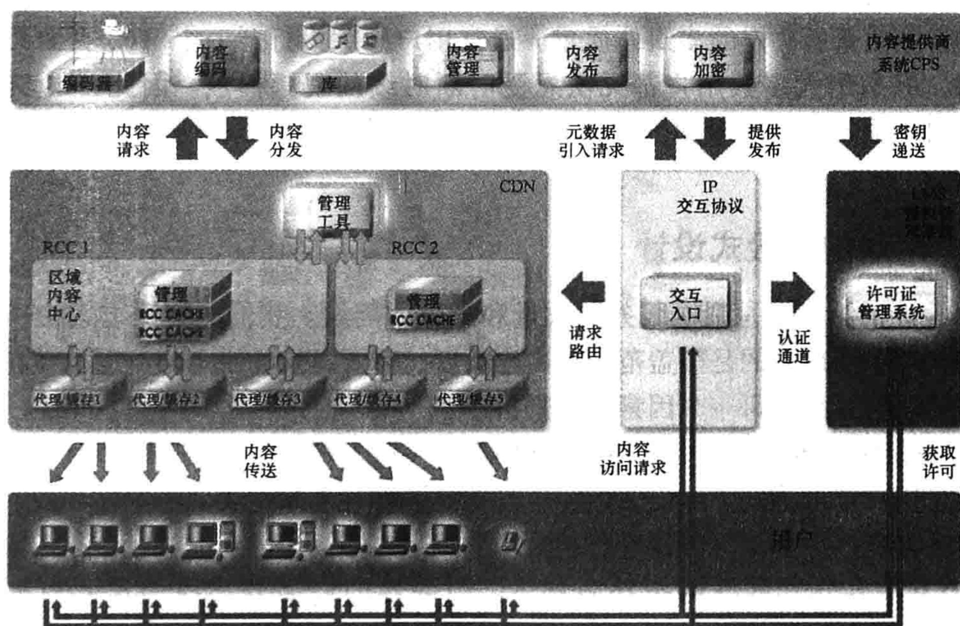


图 13.1　iTVP 平台组件

从组织结构上说，内容分发主要有三种实体参与。首先，需要有大量的独立内容提供商来构成多媒体内容的资源；第二种实体是 CDN 管理员，负责监控系统运行，这个角色由 Poznan 超级计算机和网络中心（也是波兰全国范围内所有光纤主干网的管理者）来执行，称为 PIONIER；第三，大量独立的服务提供商的参与，保证了其用户能访问 iTVP 平台服务。

iTVP 平台原型最早在 2006 年开始进行部署，完整的功能运行阶段从 2007 年年初开始。目前该平台正在逐步增加其节目和相关服务，同时用户数量也在稳步上升。起初，内容库提供数千种内容，该数字在 2007 年底几乎增加了一个数量级。每月接收内容分发的独立 IP 地址的数量从 2006 年上半年的 10 万左右，增长到 2007 年底的 50 余万。目前，iTVP 平台有两个内容提供商：波兰公共电视台（TVP）和公共广播。播放节目包括实况转播和内容点播，其中实况转播可以同步播放也可以预先播放，或者以专为互联网平台准备的方式播放。内容点播包括实况内容的归档版。目前，内容库随着新制作节目的加入不断变大，同时也随着归档节目的数字化而继续增加。

13.4 iTVP CDN 架构

一般而言，确定 CDN 副本服务器数量和位置的问题可以通过最小化 CDN 运营成本来解决，同时也可反映出资源的需求。不过，另一个方法是最优化用户体验到的服务质量。更好的服务质量意味着需要更多的资源来进行内容分发。因此，这两个方法是相互冲突的。在多媒体内容的情况下，服务质量与其他类型的内容在定义方式上完全不同，确保一定的服务质量在多媒体 CDN 规划中会起到非常重要的作用。访问时间仅仅是服务质量的参数之一，另外一些重要因素包括多媒体播放的连续性及由视频分辨率和音频质量所决定的多媒体质量。通常用户愿意为打开一个 Web 页面而等待一段时间，但是绝大多数用户都无法忍受时断时续的视频播放。因此，很多 iTVP CDN 的设计都是在服务质量和分发成本之间进行均衡，但至少必须确保最低限度的服务质量。

13.4.1 两层的分级式设计

CDN 设计的第一步是副本服务器数量和放置的问题。我们的解决方案由一系列因素构成，其中最重要的是覆盖范围、内容类型、需要提供的服务类型以及潜在的传输网络拓扑。下面针对每一个因素进行剖析。

iTVP 平台用来向全国范围的用户提供服务。如前文所述，被选择的内容对于互联网的任意用户都可访问。然而，内容分发系统被设计用来主要服务本国用户。在这个假设下，很明显，副本服务器应当被放置在那些端用户高度聚集的地带（如大城市），并且在内容分发网络的拓扑限制下，服务器的地域分布应当与人口分布相匹配。

iTVP 所提供的内容的特性，即多媒体的直播内容和点播内容，确定了分发成本和存储成本。大多媒体文件的传输需要占用源服务器和副本服务器之间的大量带宽，以及副本服务器和端用户之间的大量带宽。将副本服务器放置到用户附近能够极大地降低与带宽相关的分发成本。然而，此方法却增加了副本服务器的数量，并会导致 CDN 内的分发成本和存储成本增加。因此，在确定源服务器和副本服务器之间的距离时，应当保证分配带宽、分发带宽和存储成本之间的均衡，以使整体成本最小化。

除非文件下载模式对于给定内容可行，并且用户选择了文件下载模式，否则需要点播的内容就通过流模式分发给用户。内容流技术降低了终端系统的访问时间和存储需求，但是对网络提出了严格要求，如可用带宽和延时抖动问题等。因此，具有一定服务质量的边缘分发，如由处于主干网边缘的服务器进行的内容分发，在技术上还难以实现，并在经济上也不划算。一个能够避免以上内容流问题的最优方式就是将广域网排除在分发路径之外，即使用放置在本地接入网的服务器来将内容以流的方式直接传给端用户。这样的方式能够和最终的接入运营商（如 ISP）一起协同实现。ISP 参与内容分发的直接目的在于确保用户具有更好的服务质量，并降低输入流量的带宽成本。

以上的讨论导致了一个两层结构的分级式 CDN 架构供 iTVP 平台使用。CDN 架构的组件如图 13.2 所示，在下面的内容中将逐步介绍这些组件。上层的节点是副本服务器（称为缓存），放置在全国所有的战略性地点。下层节点是代理服务器/缓存，被放置在不同的 ISP 网络中，并直接向端用户提供服务以确保满意的服务质量。上层节点的引入能够降低与带宽相关的分配成本和源服务器负载。在两级架构中，通过内容提供商的数据库（如源服务器）获得的内容将被送至缓存节点，再从缓存处被送至代理服务器/缓存并继续传递至端用户。我们随后会看到，还有其他可用的分发路径。为了使路径之间彼此区分，我们定义 CDN 中所有从源服务器至代理服务器/缓存的内容传输称为内容分配（content distribution），从缓存/代理服务器到端用户的这最后一步内容传输称为内容分发（content delivery）。这样，就可以通过上述定义将分配成本和分发成本区别开来。

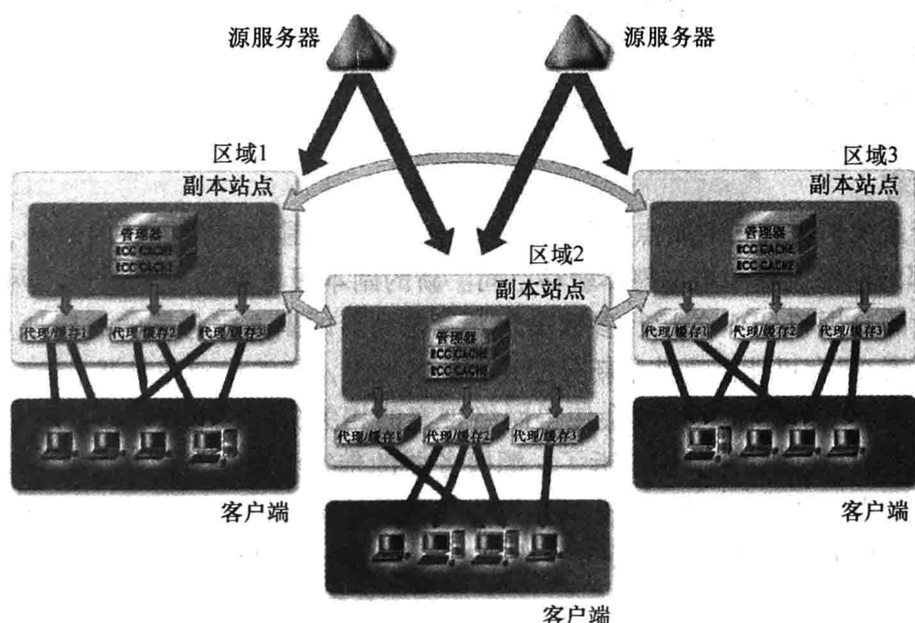


图 13.2　iTVP CDN 架构

内容一般是以流模式传递给端用户。然而，流仅仅发生在代理服务器/缓存到端用户的这一段，并不包括所有分配路径上的实时传输，从 CDN 到代理服务器/缓存的内容点播的分发以文件下载模式执行。下一节将在描述内容分配和分发规则的同时来验证该方法。因此，我们并未使用可扩展的内容流技术，而考虑的是单播内容分发模型中与服务器数量相关的系统开销，具体而言，与网络带宽相关的分配开销随着副本服务器的数量增加而降低，内容分发所需要的服务器带宽并不取决于服务器的数量，而存储成本随着服务器数量增加而增加。

13.4.2　CDN 节点放置

对于副本服务器的托管而言，城域网凭借其连接性和带宽的可用性是一个良好的潜在放置地点。一台副本服务器可以托管多个缓存服务器，它们都被与此副本服务器

站点相连的代理服务器/缓存和与其他副本服务器放置点的节点视为一个更高级的缓存池。因此，从现在开始我们在 iTVP 术语表中使用“副本服务器”，或者“区域内容中心（RCC)”来指一个作为缓存集群的副本服务器站点。在一个副本服务器处放置多个缓存能够增加可用资源（主要指服务器带宽和存储能力)，并提高服务器的可靠性。

代理服务器/缓存的数量取决于与 CDN 协作的 ISP 数量，每一个 ISP 可以拥有多个代理服务器/缓存，这取决于 ISP 的网络大小（指潜在用户的数量)。无论网络大小，为可靠性起见，一般推荐至少需要两个代理服务器/缓存。为了最小化分配成本，代理服务器/缓存与核心网络拓扑架构中距离最近的副本服务器相连。

一个副本服务器站点和所有与该站点相连的代理服务器/缓存组成了一个区域。区域的运作并不与任何中心实体协同工作。每一个区域都有一个区域管理器并具有一定的自由度。然而，对区域进行管理时需要意识到其他区域的存在，并彼此之间相互协作来执行内容分配任务。内容可以从一个区域被分配到另外一个区域，但是只能在这个分层架构的高层中（如缓存之间）进行。低层节点之间的协同只能在一个 ISP 网络的内部节点中进行，另外，代理服务器/缓存只能通过自己所处区域内的副本服务器获得内容。

随着系统扩张，新的站点需要加入以便在全国范围内进行更好的服务，并均衡 CDN 内部的网络流量和增加整体缓存空间。新的副本服务器的放置地点由网络拓扑、网络流量的分析和 CDN 的运行参数决定。考虑到新的副本服务器的放置地点，一些代理服务器/缓存可能需要被重新分配到新的区域去，新的代理服务器/缓存也有可能加入。服务器站点数量的增加会使得网络带宽的相关分配成本降低，因为代理服务器/缓存和缓存之间的距离被拉近了。源服务器的带宽请求并没有增加，原因在于副本服务器站点之间彼此协作。一台副本服务器可以通过另外一台副本服务器获得内容，而不需要通过源服务器进行。新的副本服务器站点的放置部分取决于网络的拓扑结构。

13.4.3 网络层配置

内容分配是通过 PIONIER 进行的[4]，PIONIER 是一个波兰全国范围的基于密集型光波复用（Dense Wavelength Division Multiplexing，DWDM）技术的全光网络（all - optical network)。PIONIER 与 21 个城域网相连，这些城域网都是托管副本服务器的潜在地点（见图 13.3)。缓存集合包含通过千兆级以太网技术与 PIONIER 设施相连的节点，并使用 1Gbit/s 的通道专用于传输 CDN 流量。ISP 网络中放置的代理服务器/缓存通过 100 ~ 400Mbit/s 专用通道与副本服务器相连。内部 CDN 通信层和内容分发层相互连接，形成一个用于 CDN 运行的虚拟网。虚拟局域网（VLAN）使涉及 CDN 的流量和其他网络中的应用操作区别开，提供了更好的服务质量和高级别的内容分发安全性，这一点对于多媒体行业来说非常重要。



图 13.3　PIONIER 的拓扑

13.5　内容的分配和分发

CDN 的性能不仅仅取决于副本服务器的数量和位置，如 CDN 配置的问题，它还取决于内容分配方面。内容如何存储在 CDN 中，以及存储在多少可用位置的相关决定都会影响存储空间组件方面的整体成本和用户感受到的服务质量。本节解释 CDN 内容分配和分发的设计思路，读者需要具备前面章节提及的 iTVP CDN 拓扑的相关知识。

13.5.1　内容分配模式

在 iTVP CDN 中，制定内容分配规则基于以下假设：用户从其 ISP 网络内放置的代理服务器/缓存中获取内容，因为代理服务器/缓存直接向端用户提供内容，而源服务器和副本服务器都是不可见的。每一个请求的对象都必须存在于这些代理服务器/缓存中的至少一台服务器上，假设用户从一个节点处获得一个完整的内容。在内容播放过程中切换到其他代理服务器/缓存处的情况极为少见，仅仅出现在存在节点故障时。存储在一个 ISP 网络内的代理服务器/缓存处的一组对象主要由用户请求决定，与标准的缓存系统一样。因此，内容副本的数量和它们在 CDN 中 ISP 网络层的放置地点也取决于用户请求。一般来说，每一个内容都存储在一个 ISP 网络内预先定义好数量的代理服务器/缓存处。目前，该数字设置为 2，即两个节点。

iTVP CDN 的内容分配可以被视为内容缓存和内容复制的集成。内容在缓存未命

中的情况下，即在 ISP 网络内任何代理服务器/缓存处都无法获得该内容的情况下，由用户启动拉取模式来将内容分配至代理服务器/缓存处。另外，一个推送模式也可以被一个内容提供商或者 CDN 运营者启动，并在一个新内容被分发或者提供商预计对该内容的访问量会很大时使用，另外也在内容的流行度因时间变化而增加的情况下使用。推送模式下的内容分配或复制的目的在于降低缓存未命中的几率，并由此减少内容访问时间。拉取模式下的内容分配，会调整内容可用度以便匹配内容访问量。访问量大的内容会存储在大量节点处，访问量小的内容可能仅仅存在于少量代理服务器/缓存处。每一个代理服务器/缓存都使用一个存储空间，供内容缓存和内容复制同时使用。这种内容分配的解决方案允许 CDN 将不同方法的优势进行结合。

因为 iTVP 平台提供的内容特性，代理服务器/缓存无法长期存储所有被请求或者复制的内容。因此需要管理存储空间，以确保高命中率以及确保由于缺少存储空间导致的服务拒绝事件不会发生。缓存内容的替换策略可以被描述为一种改进的 LRU 算法。修改过程考虑了内容大小，因为相对于访问量不高的大文件而言，一个访问量高的小对象可以考虑被优先替换。对内容大小的考虑降低了分配中的带宽使用成本，这对于大文件而言非常重要。对于给定内容的分配模式（如缓存或复制），在内容替换的选择中并未加以考虑。因为替换过程非常消耗系统资源，需要定期执行并将存储空间释放到一个预先定义好的大小。替换频率的选择和将存储空间释放到何种程度对 CDN 的性能有极大影响。释放过多空间会增加分配成本和缓存未命中的几率，相反如果存储空间释放不够会导致服务被拒绝。

代理服务器/缓存需要从它们所在区域的高一级节点处获取内容。分配给某一给定副本服务器站点的内容被存储在至少一个缓存服务器处。为了最小化存储需求，每一个内容只能存储在一个区域内的一个缓存服务器处，这也是 iTVP CDN 中的现行策略。一个区域内位于高一层的内容副本数量如果多于 1 个，会提高服务的可靠性并增加服务器的可用带宽；但这样做无形中增加了存储成本。因此，缓存处存储的内容副本数量的选择被用于平衡存储成本和带宽成本。通过源服务器或另一个副本服务器的缓存，内容被分配到缓存服务器。CDN 区域通过推送和拉取的方式与内容分配协作，这就实现了协作式的内容复制和缓存。这种协作方式更进一步降低了源服务器的负载和分配成本。代理服务器/缓存之间的协作被限定在一个 ISP 网络内的一组节点上。一个 ISP 网络没有理由为另外一个 ISP 网络提供内容。在一个 ISP 网络内部，内容可以从一个代理服务器/缓存处分配到另外一个代理服务器/缓存处，但是这样的传输会增加代理服务器/缓存的负载，并影响端用户的服务质量。对副本服务器站点的存储空间管理方式类似于代理服务器/缓存空间管理，一种情况除外，即前者的缓存替换过程不是定期进行，而是当需要释放存储空间时才进行。

13.5.2 内容传输模式

点播内容通过文件形式进行分配，播放速度一般要比从源服务器一路到代理服务器/缓存的方式快得多。流仅仅用于从代理服务器/缓存到端用户的内容分发。直播内容（如直播电视节目）是通过内容提供商的服务器以流的方式传递到副本服务器，并

通过副本服务器中继到其他副本服务器。每一个区域内服务器中的副本通过中继的方式将流传递到代理服务器/缓存，而代理服务器/缓存接着将内容流传递到端用户。因此，直播内容的分配可以被视为应用层的组播，与 CDN 节点一起形成了一个覆盖网。对点播内容使用文件模式分配有多种好处。因为文件模式分配高传输速度的特性，使得整个内容能更快地以流的形式分配，因此允许用户能执行 VCR 之类的操作。因为流仅仅发生在 ISP 网络内部，因此提供给用户的服务质量不会取决于 CDN 内的网络条件。另外，文件模式分配的流量模型要比流模式简单得多，能够极大地简化网络带宽管理。有两种机制可以进一步减少内容访问时间。首先，使用 CDN 分层机制的分配可通过直通或者管道方式进行，即从源服务器到一台副本服务器（或两台副本服务器之间）的传输与副本服务器到代理服务器/缓存之间的传输是并行完成的。直通式分配限制了分配路径长度对于分配时间的影响。其次，代理服务器/缓存到用户的流可以通过所谓的快速启动模式进行，其中视频文件并不是一定要等到全部传完后才能播放，而是当达到所需最小信息量时即可启动流，这个时间要远小于整个文件传输完成所需要的时长。从用户的角度来看，快速启动模式使得内容分配所需的时间（即启动内容流的时间）与内容的大小无关。

13.5.3　瞬时拥塞处理

CDN 系统中的一个重要功能就是具有处理瞬时阻塞的能力[16]。发布一个被大量用户感兴趣的内容很容易地导致用户请求访问量在短时间内迅速升高，并因此使同时在线的用户数量激增，这样的情形对于用户感兴趣事件的实况转播而言很常见。一般而言，这种内容的传输需要提前规划，同时内容分配也需要在事件开始前就执行。如果事先没有料到用户请求数量的激增，那么开始的少量请求会使内容被分配到大量的代理服务器/缓存中去，以源服务器为根节点的内容分配树将快速建立起来。因为管道内容传输和快速启动模式的原因，从代理服务器/缓存开始的内容流可以在很短时间内迅速启动。后继的用户请求会被分散到所有可用的代理服务器/缓存，此时该内容已经可以从代理服务器/缓存中获取。区域管理器需要具备很高的请求处理吞吐能力，因为所有关于路由的信息都可在本地获取。

13.6　用户请求路由

用户请求路由是 CDN 的一个功能，用于完善 CDN 拓扑和内容分配策略。至于选择哪个节点能够将内容提供给用户的规则，则部分地由 CDN 拓扑和可用内容决定。一般而言，会有一个节点集合用于向特定用户提供服务。从请求路由方法的角度来说，多媒体内容在此方面需要与其他类型的内容区分开来。当流分发的网络需求确定后，分发路径参数是节点选择过程中最重要因素之一，仅次于服务器负载。这些参数

不仅包括需要传输的距离，而且还包括可用带宽和延时抖动等。在本节列出关于iTVP CDN节点的选择标准，并描述用于用户请求路由的机制。

13.6.1 节点选择标准

对于一个通过IP地址来识别的用户，有多种标准可以用来确定由哪个节点向该用户提供服务，包括用户相对于代理服务器/缓存的位置、内容可用性、节点负载。节点选择策略首先通过主标准选择出一组能够向给定用户提供服务的节点，并通过第二个标准从该表中筛选，直至一个预先定义好的节点数量。

能够向给定用户提供服务的节点的初始集合通过用户的IP地址和IP子网与代理服务器/缓存之间的映射决定。该映射反映了某些ISP网络中用户的所在地点。IP子网与代理服务器/缓存之间的映射根据与CDN合作的ISP列表创建。除了在不同ISP网络中放置的代理服务器/缓存之外，还存在一些代理服务器/缓存能够给任意互联网用户提供服务。这些代理服务器/缓存与主干网相连或者与1Gbit/s通道的城域网相连。相对于ISP网络中被指定的节点而言，它们被称为默认代理服务器/缓存，其存在的目的是向不属于与iTVP平台相关的任何ISP中的用户提供访问iTVP的服务，并提供这些用户所在的ISP中的代理服务器/缓存中无法获取的内容的访问服务。后者的角色是在出现缓存未命中时，能够降低用户访问时间。如果用户请求的内容位于任意一台默认代理服务器/缓存上，用户请求就会被路由至此代理服务器/缓存处，同时内容分发开始转至被指定的代理服务器/缓存。因此，一般而言通过用户IP地址所选择出来的节点集包含两种服务器，即指定的和默认的代理服务器/缓存服务器。

考虑到在ISP网络中可能存在多于一台的代理服务器/缓存，节点选择的第二个标准是请求内容的可用性。这种方法能够最小化内容访问时间，并降低分发开销。内容可用性或内容定位通过在每一个区域内维护的内容目录中存储的信息决定。因此，任何内容都可以通过一份区域内目录查找表来进行快速定位。每一个区域管理器负责维护该内容目录表，同时为其他区域充当目录服务器。快速内容查找所产生的成本很大程度上与目录更新系统成本有关。然而，区域管理器会管理所有节点的运行，包括内容分配。因此，大多数内容定位更新随内容分配而进行，这样自身不会增加通信系统成本。从一个节点上移除内容所导致的缓存替换过程不会显著增加系统成本。一个中心内容目录（或中心管理器）可能发生单点失效，这被认为是此类解决方案中最严重的缺陷。为了确保可靠性，一般每个CDN副本服务器中配备有至少两个缓存服务器，能够负责接管内容目录服务器的任务和实现管理器的职责，这样就能够确保对管理区域而言至关重要的数据不会在管理系统出现故障时丢失。

在所有能够向用户提供服务的节点中，请求内容能够得到满足的节点将被选定。如果存在多于一个节点符合以上条件，那么负载最小的代理服务器/缓存（如满足最小数量的在线用户要求）将被优先选择成为指定的代理服务器/缓存。考虑到端用户

和给定 ISP 网络中确定的每一个代理服务器/缓存之间的静态路径特性彼此相似，并考虑到动态特性取决于负载，同时在线用户的数量反映了服务器负载和可用带宽。更精确地说，考虑到可靠性，可用节点的数量被缩减为大于 1 台即可，一般而言是两个。也就是说，用户同时收到两个地址来对其请求的内容进行服务。如果没有任何节点（甚至包括默认代理服务器/缓存在内）能满足用户请求，那么会出现一个缺失信号并启动内容分配。

13.6.2　请求重定向机制

区域管理器负责选择节点和确定用户请求路由。然而，用户并不与管理器直接交互。对于端用户来说，有很多独立入口能够确保用户与 iTVP 服务相连。一个用户请求通过一个入口被定向到一个区域管理器供后期处理使用，如用于代理服务器/缓存选择。IP 子网和代理服务器/缓存之间的映射转变为 IP 子网和 CDN 区域之间的映射。因此，对于一个给定用户的 IP 地址，应当存在一个区域能够为该用户提供相关服务，并且区域管理器收到请求后，能够将其转发到一个合适的区域。因为在用户获得访问内容服务时，入口充当了端用户和 CDN 之间的中介，所以区域管理器对于请求的响应会通过入口被送回。该响应可以是一个包含代理服务器/缓存地址的响应，也可以是一个因为缓存未命中所返回的“等待”响应。

13.7　iTVP CDN 性能评估

基于 CDN 报告子系统收集到的数据，可以计算一系列指标，用于对 CDN 性能和用户感知到的服务质量进行评估。首先，介绍系统配置、主要内容提供商的内容库以及在数据收集期间观测到的 CDN 负载情况。其次，通过检查下面三个方面的信息，即在不同 CDN 层和相应数据库之间分配的对象数量、低层节点所用带宽引发的分配成本以及 CDN 节点使用的缓冲空间所对应的存储成本，来衡量内容分配的成本。CDN 性能也通过用户感知到的服务质量（用请求命中率来表示）来进行衡量。

13.7.1　iTVP CDN 配置

在 iTVP CDN 的现有配置中，有两台副本服务器运行在 Poznan 和 Krakow 两地，大量的代理服务器/缓存分布在不同的城市中：Warszawa、Poznan、Gdansk、Szczecin 等城市连接到 Poznan 区域，Krakow、Lodz、Katowice 等城市连接到 Krakow 区域。iTVP 内容库的源服务器位于 Warszawa，并同时和两台副本服务器相连，如图 13.4 所示。缓存节点和默认代理服务器/缓存配置了大小为 TB 级的缓存空间，指定的代理服务器/缓存的缓存空间一般为 400GB 左右。

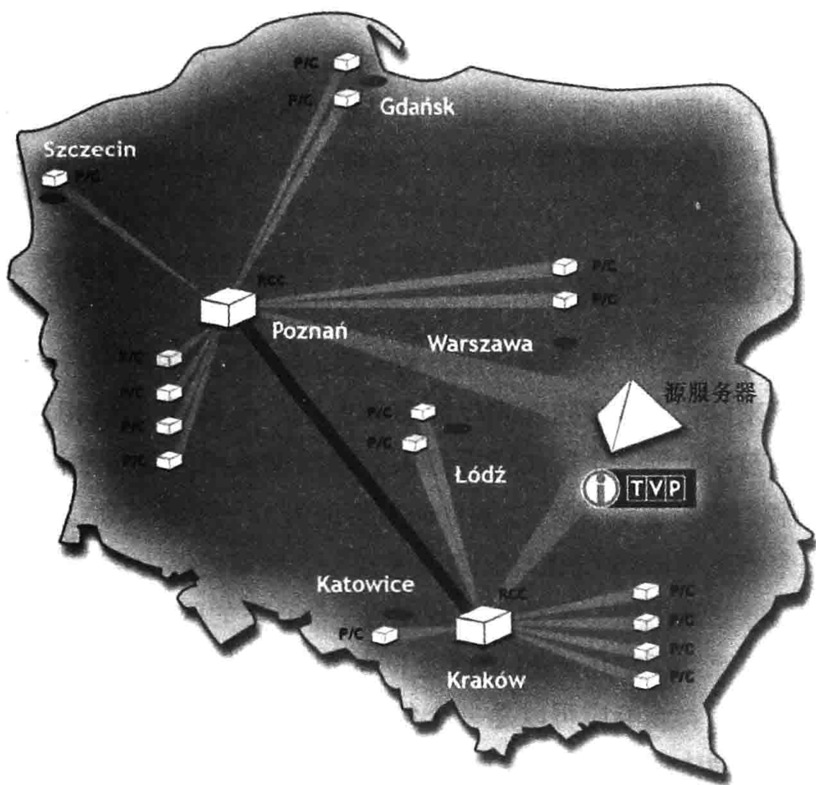


图 13.4 iTVP CDN 的配置

13.7.2 内容库特性

内容库中的媒体内容以 MPEG4 兼容格式进行编码（Window 媒体）。大多数对象都有两种质量格式可供选择，一种主要针对传输速度 28 ~ 128kbit/s 的手持设备进行编码，另外一种主要以 160 ~ 700kbit/s 的传输速率进行编码。部分内容以更高的 1.5Mbit/s 速率进行编码，需要下载的内容也以 1.5Mbit/s 的速率进行编码。每一种质量格式的编码在给定速率区间内都含有多种流，可供不同的带宽情况选用（多速率 CBR）。低质量的格式包括 3 ~ 4 个视频流，速率范围从 16kbit/s 到 91kbit/s，以及 1 ~ 3 个音频流，速率范围从 8kbit/s 到 20kbit/s。更高的质量内容格式一般包括 4 ~ 5 个视频流，速率从 121kbit/s 到 563kbit/s 不等，并包括 3 ~ 4 个音频流，速率从 20kbit/s 至 128kbit/s。图 13.5 给出了点播内容大小的累积分布函数（CDF）。大多数文件都含有数百 MB 的数据，确切地说有 46% 的数据对象大于 100MB。

iTVP CDN 每日发布的对象数量一般在几十个左右。有些新闻节目会在一天中播放多次，并且在实况直播之后会有存档版本立刻上线供点播访问。这样的内容一般仅仅在接下来的几周内被访问。因此，这样的节目分发率并不影响内容提供商的内容库中可供用户访问的内容对象的整体数量。

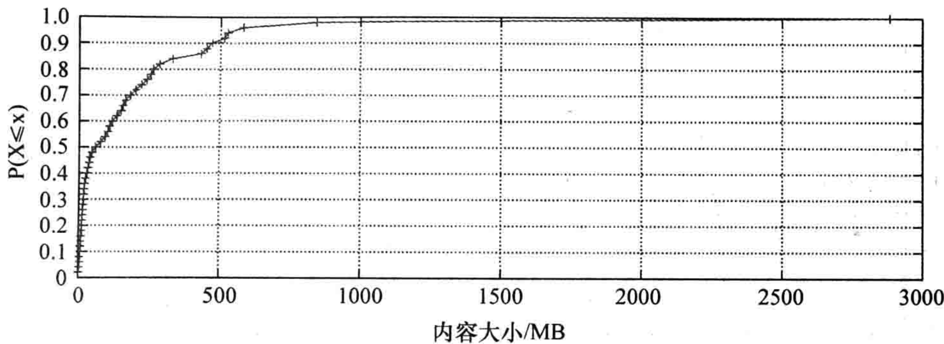


图 13.5　内容大小的累积分布函数

13.7.3　CDN 负载

多媒体 CDN 的负载可能由多个因素决定，如用户数量、用户访问量、视频内容播放的总时间及平均时间，以及端用户能接触到的数据量大小等。我们定义用户数量为发送请求并收到内容的独立 IP 地址的数量，这个数字应被视为真正用户的数量下限。不过，我们正设法提出对该用户数量更加精确的测量方法。我们观察到，用户数量的快速增长（每月以数十万计）与所提供的服务的提升成正比关系。用户访问量也在快速增长，每月超过上百万，这说明大多数用户是回头客。一个月内媒体内容播放的总时间大约在数百个播放年的数量级，每一个会话期都在 10 多分钟左右。大多数 iTVP 媒体内容都有较长的播放持续时间，这样的情况在互联网上很少见。送往端用户的数据大小主要取决于内容编码质量和可用带宽。因为这些参数都在稳步提高，因此我们每个月能给端用户提供数百 TB 的数据，这个数字还在增长。

CDN 的负载也受播放媒体内容时在线用户数量的影响。同时在线的用户数量随着时间变化剧烈，每天的不同时段和每周的不同日期都与用户数存在着一定的对应关系。与内容类型相关模式有若干种，其中的一种模式与电视和广播有关：观看电视内容的用户高峰一般在晚间来临，而广播内容的用户高峰一般出现在早上和午后。访问模式上的差异也表明，在以周为单位的尺度上，广播内容的访问量在工作日是最高的，而对于电视内容，观众的分布在周末和工作日是不同的，在周末时出现在午后的用户请求要比在工作日时多得多。图 13.6 给出了一周内同时在线的用户数量（对应于两种媒体内容类型的最大值），还给出了所有媒体内容的累计数量。访问模式也随较长的时间尺度变化，因此资源请求也呈现出随时间尺度变化的特性。

图 13.6 也显示出广播和电视节目中用户请求频率的不同特性。广播内容和基于节目的电视内容相比，对其访问更具有连续性。当把用户请求分布同随时间变化的端用户接收数据量分布相比时，各类内容访问模式的不同是显而易见的。图 13.7 显示了传输数据的累积量与一天中各时段最大值的相对关系。我们发现，累积数据量在早晨的增长曲线要比一天中平常时间的数据量增长的更为平滑，这是因为每天清晨大多数用户都会收听广播。另外，在晚上大多数用户会选择收看电视，此时需要传输更多的数据。斜率的变化在周末出现得更早，因为用户在周末要比平日更早收看电视节目。

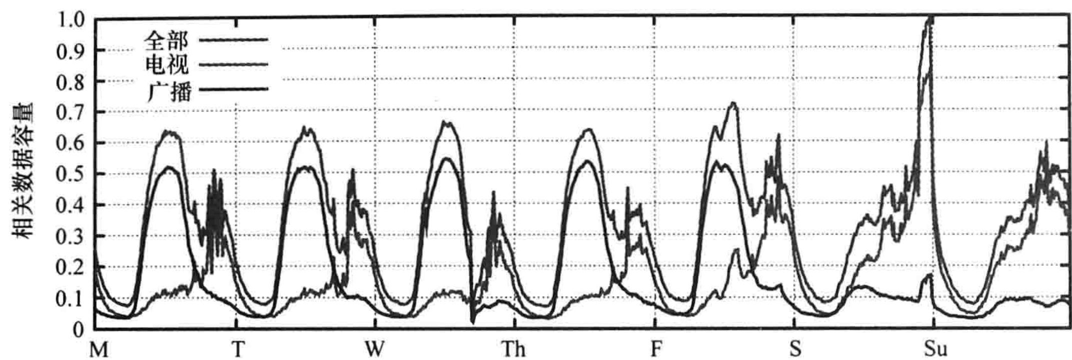


图 13.6　在线用户数的变化

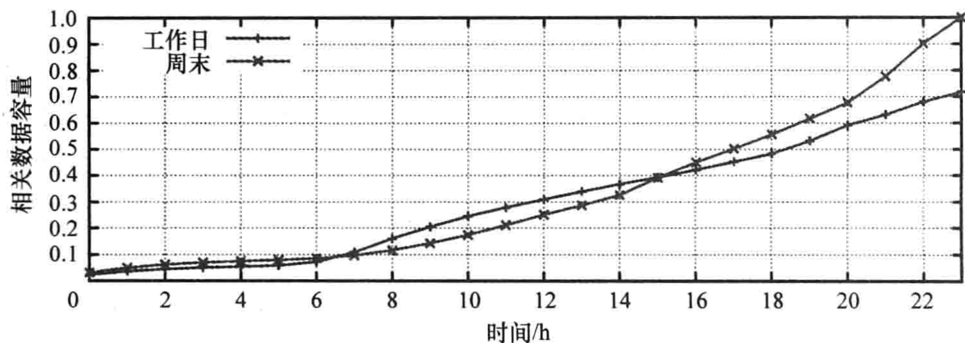


图 13.7　数据在不同时间的累积传输量

13.7.4　内容分配性能

CDN 性能通过两组量进行评估，第一组测量与资源使用率相关的系统开销，并包括分配带宽、分发带宽和存储使用率；第二组测量用户感受到的服务质量。

我们利用 CDN 的分层机制，通过在不同层的节点数据之间的比较（具体指从上一层传输到当前层的数据量和从当前层传输到下一层的数据量之间的比较），来对 CDN 不同层之间的效能进行评估。因为数据流不仅仅发生在 CDN 的相邻层，也可以在高层中平行进行，因此首先对副本服务器之间的协作优势进行评估，这可以通过计算源服务器的实际负载与无副本协作情况下的源服务器负载得到。图 13.8 给出了每天从源服务器获得的数据量和从源服务器传输到副本服务器的数据量之比，以及每天从源服务器获得的数据量和从另外一台副本服务器传输到该副本服务器的数据量之比（即在没有副本服务器协作的情况下从源服务器获得的数据量）。对几个月的数据做平均计算之后得到的数值大约是 0.55，这表明通过源服务器传输的数据量和副本服务器之间交互的数据量是差不多的。该比率偶尔能够接近 1，说明大部分内容是通过源服务器获得。一般而言，以下情况较为常见：当少数访问量低的数据对象通过一台副本服务器从源服务器处获得时，因为访问量低，它们随即不再被传输给其他副本服务器。基于这个事实我们估计，在没有副本服务器协作的情况下，源服务器的负载有可能翻倍。这个数字验证了副本服务器之间的协作。并且，我们推测副本服务器数量

的增长不应该对源服务器的负载有明显影响，因为所有副本服务器都是通过全网状网的形式相连，并且每一个区域内的负载都保持在合理水平线上。此处的合理水平线是指在每一个 ISP 网络内的代理服务器/缓存都能够在几个小时内缓存 ISP 用户所请求的所有内容，也就是说，在此段时间内缓存内容相对稳定。在一个拥有 n 个副本服务器的系统中，有副本协作的源服务器负载与没有副本协作的负载之比应该在 1：($n-1$) 左右，这个比率是以副本服务器间增加的带宽使用成本换来的。

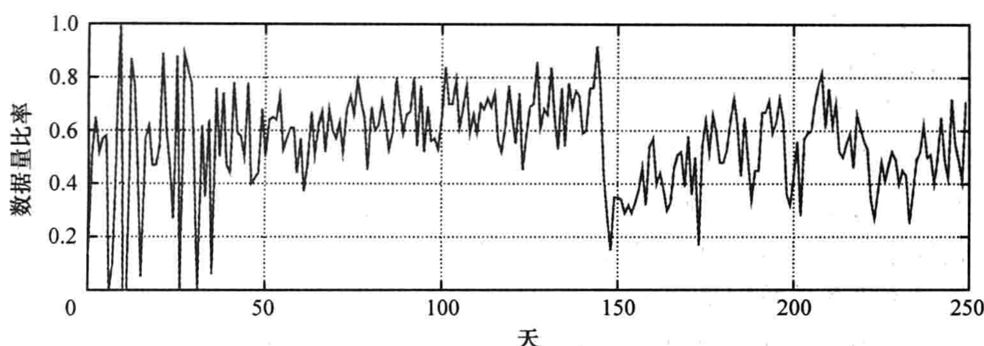


图 13.8　副本服务器协作对于源服务器负载的影响

接下来，我们检查每天的数据吞吐量（抵达副本服务器和来自副本服务器），数据吞吐量通过在给定的副本服务器上，对所有缓存求和得出。图 13.9 给出了每天每台副本服务器获得的数据吞吐量与给定的某台副本服务器对它所在区域内的所有代理服务器/缓存所传输的数据吞吐量总和之间的比率。以几个月的周期做平均之后得到的数值大约是 0.13（对于 Poznan 副本服务器）和 0.12（对于 Krakow 副本服务器）。两个数值处于同一数量级，差别可以用加载在 Poznan 副本服务器上的较高负载来解释。因为有了 CDN 分层机制中高层的存在，这个进出数据量比率使得我们能够预测源服务器的负载降低情况。如果代理服务器/缓存需要直接从源服务器处获得内容，那么该服务器的负载会高出 8 倍左右。因此，我们得出结论，CDN 的高层节点对于减轻源服务器负载非常有效。副本服务器的负载取决于它所在区域内代理服务器/缓存的数量，二者是线性关系。因此，给定区域内用户数量的增长，会导致代理服务器/缓存的数量增长，同时也会导致另外一台副本服务器的加入，并拆分已有区域。

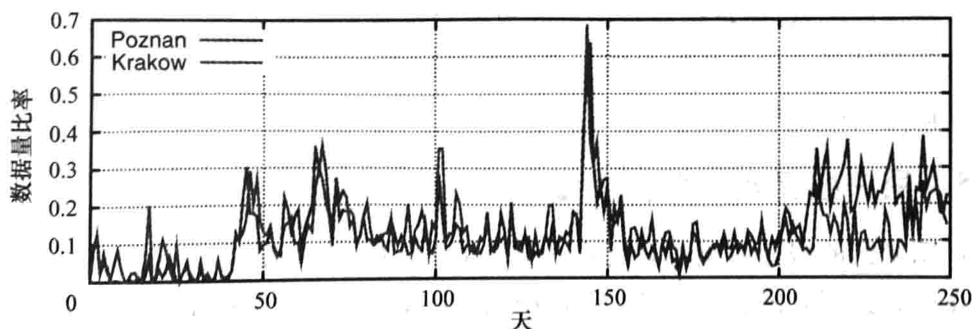


图 13.9　副本服务器收发数据量的对比

针对 CDN 分层架构中处于低层的代理服务器/缓存，我们对节点的进出数据量进行了一个类似的比较。不过，在这种情况下还需要考虑到一个事实，即用户经常不会看完全部的内容，因此往往只有部分内容会传递给用户，同时整个内容被存储在代理服务器/缓存处。因此，除了被代理服务器/缓存传输给用户的数据量之外，同时还通过对所有用户访问的对象（无论哪一部分真正被用户看过）的大小求和，计算其理论上限。代理服务器/缓存获得的数据量和用户每天获得的数据量之比大约是 0.04。这说明，代理服务器/缓存传输的数据比它们获得的数据高出 25 倍左右。代理服务器/缓存获得的数据量和数据传输的理论上限之比大约不到 10 倍。代理服务器/缓存负载与用户数量呈线性关系，因为内容通过单播的方式传给用户。参照从源服务器到 CDN 的数据传输，也考察了代理服务器/缓存传输的数据量，为了估计源服务器的负载，我们将考虑设有 CDN 时的情况（即不存在缓存代理的情况）CDN。结果表明，当一个代理服务器/缓存的用户全部定向到源服务器时，源服务器传输的数据量大约是它传给副本服务器数据量的 50 倍。

为了评估内容缓存的效用，也计算了在不同时间尺度下位于 CDN 不同层的每个内容被访问的平均次数，该值较高原因有可能是因为资源不足，如较少的存储空间、可用性，或因为低效的存储空间管理，如内容替换策略等。在几个月内对源服务器的访问次数做平均，可知每个内容的访问次数大约是 1.02，这表明绝大多数对象仅仅在源服务器上被获取过一次。在每一个副本服务器上，内容访问的平均次数和这个数量相差无几。对于代理服务器/缓存而言，该数值会高一些，在几个月的时间尺度上做平均的结果在 1.2 左右。当降低时间尺度再做平均时，该数值迅速趋于 1，说明就算一个内容从副本服务器上被获取的次数超过一次，彼此之间也会相隔数天或者数周。上述分析表明，CDN 具备了高效的存储空间供负载使用，并且系统管理良好。

源服务器负载和网络带宽使用率的下降所付出的代价是对存储空间的更高要求。我们将一段时间内某节点存储的数据量与该节点传输的数据量进行了比较，借此对 CDN 运行中与存储相关的开销进行了评估。存储空间使用率通过一段时间内一个节点存储的所有对象的大小之和计算。对于副本服务器而言，存储和传输之比与获取和传输之比非常接近，因为每一个内容的访问数略高于 1。因此，估计日常情况下每一个副本服务器传输的数据要 8 倍于它存储的数据。每天代理服务器/缓存的存储数据量与传递给用户的数据量之比大约是 0.4，如果用户在每次访问某内容时都要播放全部内容，代理服务器/缓存传递给用户的数据量大约要高出 8 倍，这就会使存储与传输的比值相对较低。以上数据表明，采用两层体系架构组织起来的大量 CDN 节点可以增加存储空间的可用数量。

13.7.5 用户感受到的服务质量

用户感受到的服务质量表现为命中率，即能被至少一个 CDN 节点服务的用户请求数。在内容分配方面，命中率与 CDN 性能直接相关。内容播放的质量对于用户感知质量的影响非常明显；然而，它主要取决于从代理服务器/缓冲到用户之间的网络

连接状况。播放质量难以通过客观方式进行量化，因此我们使用内容访问时间对 CDN 的效能进行评估。访问时间取决于用户请求是否得到响应。如果得到响应，那么响应时间更多地取决于管理节点与用户之间的网络连接，而非 CDN 的整体效能和区域管理器的负载。如果没有得到响应，那么直通式的内容分配和快速启动模式会使分配时间和分配路径与内容对象大小无关。因此，可以将用户请求分成“命中”或者“缺失”来评估内容访问时间。命中率在用户层上定义，并且往往高于从代理服务器/缓存处获得的命中率，原因是用户可以从默认的代理服务器/缓存处直接访问内容，而传输给指定代理服务器/缓存的内容分配是在响应请求时才启动的。一般而言，从 CDN 中观察到的命中率在 0.9 以上，因此我们有理由认为绝大多数的用户感知到的访问延时微乎其微。

13.8　未来研究方向

事实上，虽然 iTVP CDN 是一个在大地域尺度上已经完全处于自主运营中的成熟网络，未来研究中仍然有若干值得关注的方向，这其中有些与内容分发直接相关，有些则关注于提供给用户的服务集以及这些服务集会给 iTVP CDN 带来什么样的影响。

其中的一个挑战将会是监控系统的扩展情况，并按需对 CDN 进行扩容。进行决策时，可以根据资源使用情况和用户感受到的质量来添加新的副本服务器和选择放置地点，并对已有区域进行划分。也就是说，难点在于监控整个系统，并根据系统架构来确定系统扩展的规则，当然该系统架构在设计时应确保 CDN 的高扩展性。

目前，iTVP CDN 中的内容分配主要通过拉取模式进行，每一个节点的缓存内容主要由用户请求和对请求的响应来决定。推送模式的内容分配机制主要用于高访问量内容发布之后的实时传输。我们将研究使用推送模式在更大的程度上作为拉取模式的补充的效果。前面的结果显示了，CDN 的负载以天或周为单位变化，因此资源使用情况也随着这样的时间尺度发生变化。我们计划将那些未使用的资源用于内容分配，以提升用户感知到的质量。特别地，在资源可用度很高的情况下，在用户访问之前就规划好内容的分配。

iTVP CDN 本来是被设计为一个供大量独立内容提供商使用的平台。我们期望，未来不仅仅内容提供商的数量会增加，端用户也能够扮演内容消费者和内容提供者的双重角色。也就是说，在大尺度上用户组之间的内容交换会成为未来的一个潜在需求，这些变化将能够决定未来的研究方向。我们将研究是否需要实现一个混合式系统，其中 CDN 和 P2P 结合在一起，就如同 Dong 等人[9]和 Xu 等人[21]所提出的那样。这样的组合是可行的，并可能具备一定优势（如在 ISP 网络的内部）。然而，直接将 P2P 类型的分配机制应用到多媒体流会带来一定的挑战，有很多问题需要解决，如端用户可用带宽的非对称性问题。

构建 iTVP 平台的一个主要目标就是给端用户提供交互式的访问服务。交互式服务逐渐被引入，一些服务（如实时流传输中的时间条拖动或者个人视频点播）都能以个性化的方式向用户提供，并对 CDN 的资源使用带来重大影响。因此，我们计划

对交互式访问模式下的 CDN 设计问题进行研究。除了这些新服务，我们也计划扩展终端设备的范围，使得用户能更加方便地访问 iTVP 服务。具体而言，我们考虑引入多种多样的手持设备，如掌上电脑或手机。此类设备不仅需要我们调整内容的形式和格式，也需要考虑内容分配的新局面，即用户的移动性。因此，我们将研究用户从一个终端设备迁移到另外一个终端设备的移动性，例如用户有可能会从手机转移到另一个更高质量的设备。

13.9 写给使用者

我们相信，多媒体 CDN 的发展会为网络感知服务铺平道路，并能使其通过宽带 IP 网络无缝对接到端用户。很多面向消费者的大规模应用对于网络性能要求很高，需要在全网进行分发，因此需要对网络架构进行详细部署和周密策划，以达到所期望的性能和质量要求。我们预计，多媒体 CDN 开发的下一步工作将表现为从内容分发转变为网络层的高级服务分发。这一变化使得应用服务的透明配置成为下一代 CDN 的主要目标。未来的应用服务将使用内容资源以及端用户输入，这些都将通过动态、高度分布式和多功能的工作流来进行处理。因此，应用服务将分发个人化和交互性的多媒体体验。这种方法会使多通道的内容分发和不同多媒体环境中的系统集成变为可能，并使下一代多媒体应用能够根据端用户的性能来规划内容。这样的开发场景将会在很多研究领域带来挑战，如资源管理和内容分配、工作流程的建立和规划、任务和相关性管理、服务挖掘、服务监控以及网络层集成等。我们期望在不远的未来，这样的环境不仅能够由市场趋势来驱动（如高清视频、网络交互游戏或者虚拟工作环境等），而且也通过端用户对任何时间和地点访问服务的期望来驱动。

13.10 总结

本章提出了 iTVP 平台，该平台的目标是通过宽带 IP 网络向用户提供大规模的视频内容直播和点播的分发服务。iTVP 平台的核心组件是一个 CDN 网络，设计一个 IP 网络上分发多媒体内容的 CDN 需要考虑到多媒体内容本身的特性和内容传递模式，即内容流的特性。我们描述了影响设计的若干要素，并提供了相对应的 CDN 架构以及 CDN 节点的放置规则、内容分配、用户请求路由机制等。因为 iTVP 的 CDN 是一个真实的系统，部署并运行在真实环境中，因而使得我们能有机会验证所提出的设计方案。我们分析了两层分级架构的优势和成本。在实验中发现，iTVP CDN 能有效降低源服务器的负载，并高效地利用网络带宽和存储空间，同时也能够确保端用户快速访问所需内容。我们也探讨了 CDN 的可扩展性，因为 iTVP 本身就是一个高度动态的系统。随着内容提供商资料库的持续增长和节目日益具有吸引力，iTVP 平台的新用户随之不断增加。因此，我们对资源使用率和用户感知到的服务质量进行了持续监控。利用系统收集到的数据，可以对内容访问模式进行详细的分析，从而能够对 CDN 的功能设计进行指导，并使资源得到更加有效的利用。最后，我们讨论了这个方面未来的若干研究方向。

参考文献

[1] Aggarwal, C., Wolf, J.L., Yu, P.S.: On optimal batching policies for video-on-demand storage servers. In: IEEE Conference on Multimedia Systems, pp. 253–258 (1996)

[2] Almeida, J.M., Eager, D.L., Ferris, M., Vernon, M.K.: Provisioning content distribution networks for streaming media. In: IEEE INFOCOM, Vol. 3, pp. 1746–1755 (2002)

[3] Almeida, J.M., Eager, D.L., Vernon, M.K., Wright, S.J.: Minimizing delivery cost in scalable streaming content distribution systems. IEEE Transactions on Multimedia **6**(2), 356–365 (2004)

[4] Binczewski, A., Meyer, N., Nabrzyski, J., Starzak, S., Stroinski, M., Weglarz, J.: First experiences with the Polish Optical Internet. Computer Networks **37**(6), 747–760 (2001)

[5] Cahill, A.J., Sreenan, C.J.: An efficient cdn placement algorithm for the delivery of high-quality tv content. In: 12th annual ACM international conference on Multimedia, pp. 975–976 (2004)

[6] Cahill, A.J., Sreenan, C.J.: An efficient resource management system for a streaming media distribution network. Interactive Technology and Smart Education **3**(1), 31–44 (2006)

[7] Cranor, C.D., Green, M., Kalmanek, C., Shur, D., Sibal, S., der Merwe, J.E.V., Sreenan, C.J.: Enhanced streaming services in a content distribution network. IEEE Internet Computing **5**(4), 66–75 (2001)

[8] Czyrnek, M., Kusmierek, E., Mazurek, C., Stroinski, M.: Large-scale multimedia content delivery over optical networks for interactive TV services. Future Generation Computer Systems **22**, 1018–1024 (2006)

[9] Dong, Y., Kusmierek, E., Duan, Z., Du, D.H.: A hybrid client-assisted streaming architecture: Modelling and analysis. In: The 8th IASTED International Conference on Internet Multimedia Systems and Applications (2004)

[10] Gao, L., Zhang, Z.L., Towsley, D.F.: Catching and selective catching: efficient latency reduction techniques for delivering continuous multimedia streams. In: ACM Multimedia, Vol. 1, pp. 203–206 (1999)

[11] Griwodz, C.: Movie placement in a hierarchical CDN with stream merging mechanisms. In: N. Venkatasubramanian (ed.) SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), pp. 1–15. SPIE (2004)

[12] Guo, L., Chen, S., Xiao, Z., Zhang, X.: Disc: Dynamic interleaved segment caching for interactive streaming. In: 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05), pp. 763–772. IEEE Computer Society (2005)

[13] Guo, Y., Sen, S., Towsley, D.: Prefix caching assisted periodic broadcast: Framework and techinques to support streaming for popular videos. In: IEEE International Conference on Communications, Vol. 4, pp. 2607–2612 (2002)

[14] Hu, A.: Video-on-demand broadcasting protocols: A comprehensive study. In: IEEE INFOCOM, Vol. 1, pp. 508–517 (2001)

[15] Hua, K.A., Cai, Y., Sheu, S.: Patching : A multicast technique for true video-on-demand services. In: ACM Multimedia, pp. 191–200 (1998)

[16] Jung, J., Krishnamurthy, B., Rabinovich, M.: Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In: International World Wide Web Conference, pp. 252–262 (2002)

[17] Kusmierek, E., Czyrnek, M., Mazurek, C., Stroinski, M.: iTVP: Large-scale content distribution for live and on-demand video services. In: R. Zimmermann, C. Griwodz (eds.) Multimedia Computing and Networking SPIE-IS&T Electronic Imaging, Vol. 6504. SPIE (2007). Article CID 6504-8

[18] Kusmierek, E., Du, D.H.C.: Proxy-assisted periodic broadcast for video streaming with multiple servers. Multimedia Tools and Applications, Online First (2004)

[19] Ramesh, S., Rhee, I., Guo, K.: Multicast with cache (Mcache): An adaptive zero delay video-on-demand service. In: IEEE INFOCOM, Vol. 1, pp. 85–94 (2001)

[20] Su, Z., Katto, J., Yasuda, Y.: Dynamic replication of scalable streaming media over content delivery networks. In: Communication and Computer Networks (2004)

[21] Xu, D., Kulkarni, S.S., Rosenberg, C., Chai, H.K.: Analysis of a CDN-P2P hybrid architecture for cost-effective streaming media distribution. Multimedia Systems **11**(4), 383–399 (2006)
[22] Yang, M., Fei, Z.: A model for replica placement in content distribution networks for multimedia applications. In: IEEE International Conference on Communications, Vol. 1, pp. 557–561 (2003)

第 14 章　移动 CDN 中的信息发布

Nicholas Loulloudes，George Pallis 和 Marios D. Dikaiakos

14.1　引言

随着近些年来无线访问技术以及移动终端技术的发展，移动网络正在成为互联网的一个重要的组成部分[1, 46]。目前被广泛部署的移动网络使得新的应用具备了移动性，同时使有线 Web 应用拓展到移动终端。移动无线网络提供了一个丰富、动态、交互性的服务，如 GPS 导航信息、移动电视、车辆交通信息和面向定位服务。上述这类服务的正常运行是建立在快速、高效的数据传播技术基础之上的，该技术可以减少网络流量和移动用户的平均等待时间。

在有线网络中，网络性能通常可以通过增加额外带宽这种低成本的方法来得以提升。但是这种方法对于移动网络设施却不适用，因为这些移动网络都工作在固定的频段，而且由于电磁干扰的原因，数据传输速率存在着内在的局限性[46]。当移动用户开始使用对带宽敏感的应用服务时（如流媒体等），这个问题会变得更加严重。在这种情况下，缓存和预取是可行的解决方案。值得一提的是，这些方法已经在有线网络中被大量使用，通过将流量从过载的内容提供商处迁移到内容消费者较近的地方，从而实现带宽资源的优化[43]。尽管这些方法具备若干优点（如保留网络资源与减少延迟等），但动态内容和资源饥渴型应用（如多媒体应用）的发布仍是一个具有挑战性的问题。

内容分发网络（CDN）通过将用户所需内容移至所谓的互联网边缘，使其与用户更加贴近来解决上述问题[45]。关于 CDN 的介绍可以参看本书第一部分的相关内容。迄今为止，对于有线网络的 CDN 技术已有了大量的研究[27, 32, 40]，但是对于移动网络的研究还很少[1, 49]，这是由于前些年大多数移动终端访问互联网的能力有限。但是随着新的蜂窝网络技术（如 3G）和无线技术（如 Wi - Fi）的出现，形势已经发生了变化，移动终端访问互联网的服务和其他数据服务的速度已经能够与传统的有线访问速度相媲美[46]。先前的一些研究[6, 48]表明，协作式缓存技术在移动网络环境中可以改善网络的性能和信息发布效率。所以我们相信，CDN 设施可以提供一个可扩展且划算的机制，来加速在移动无线网环境下的信息发布[45]。但是，移动无线网络设施代表的是一个与传统的有线网络完全不同的信息媒介，具体表现在使用的访问设备、内容可用性、带宽以及用户成本等方面。所以，典型的 CDN 不能使移动无线网络得到增强，因为传统的 CDN 并没有将移动无线网的特点考虑进去。于是，我们在此定义了移动 CDN：在移动无线网络设施中发布内容的代理缓存服务器覆盖网。特别地，CDN 可以为开发新兴的移动计算技术提供一个新的平台。

本章的主要内容是介绍移动 CDN 所面临的挑战和现状，讨论移动无线网络设施的发展，以及研究信息发布在移动 CDN 中将如何改善的问题。

本章的其他部分内容将安排如下：14.2 节介绍移动 CDN 的必要性，14.3 节描述移动 CDN，14.4 节介绍移动 CDN 中的无线网络设施，14.5 节讨论有哪些现有的媒介可以在移动 CDN 中采用，14.6 节提供一些移动 CDN 的具体实现及相关实验情况，14.7 节讨论未来的研究方向，14.8 节进行总结。

14.2 动机

移动互联网，其定义为通过移动设备无线访问 Internet 上的数字内容，由于大量用户的使用而迅速发展。最近的研究表明，日本的移动互联网用户数已经超过有线互联网用户数[4]。可以说，移动用户在哪里，移动互联网就在哪里；或者说用户不论在何时何地都期望使用移动设备上网，获取所需信息。

如今，进军移动互联网的内容提供商在不断地增加。例如，在汽车工业里已经将此类移动技术引入到汽车设备中，用来给司机提供准确的导航和交通辅助（交通状况如交通事故情况、拥堵情况、路况信息及临时绕行信息等）。同时，这些设备也可以通过多媒体数据来提醒司机注意应对紧急情况（如火灾、地震、恐怖袭击等）。尽管在大多数情况下一个简单的文字信息已经足够，但是多媒体形式的信息如事故的图片、录像（或者前方危险状况的视频）使得司机可以更准确、更直观地获取信息，从而采取必要的措施。另外，银行业已经意识到移动互联网为发展自助银行业务所带来的商机。如今，许多移动手机用户已经可以通过他们的手持设备来享受这种服务。另外，移动互联网也给娱乐业带了新的思路，如在线音乐、在线图书、在线电影以及在线游戏。例如，旅行者或者上班族可以在等车的时候用他们的移动设备（如 PSP）玩游戏，也可以与另外一个地方的朋友聊天。

上述例子的实现不但需要先进的无线网络技术和支持设备，同时也需要支持高效的信息发布技术、可升级和可拓展的互联网设施。从技术的角度来说，目前先进的无线网络技术（如 3G、GPRS、DSRC㊀）已经可以确保移动设备访问互联网和其他数据服务的速度与传统的有线网络平分秋色。

CDN 设施[45]提供了一个可升级且节省成本的机制，来加速在有线网络环境下的信息发布。但是，由于未将移动性考虑在内，常用的 CDN 设施并不适合来增强移动互联网的性能。移动用户需求的变化不仅随着请求内容流行度的变化而变化，也与用户的移动性有关。每个用户的需求都与相应的内容、时间和地点有关。为了满足移动用户，服务器覆盖网必须位于无线网络提供商的基站附近。

因此，还需要重新考虑 CDN 架构来满足移动用户的需求。移动网络的一个特性就是资源缺乏；对于移动设备的小巧尺寸而言，存储能力、计算能力和电源续航能力上都存在着诸多限制。例如，当父亲用 Wi-Fi 摄影机给他正在海滩玩耍的三岁孩子

㊀ 即专用短程通信技术，见 http://www.leearmstrong.com/DSRC/DSRCHomeset.htm。——原书注

拍录像时，他会希望将视频上传至服务器，以便可以腾出空间来拍摄更多的照片或录像。不幸的是，到目前为止，传统的 CDN 还不支持将用户内容上传至代理缓存服务器（surrogate server）。实际上，CDN 的发布模块负责决定哪些内容可以由代理缓存服务器复制保存。鉴于此，移动 CDN 中的代理缓存服务器应该提供面向用户的服务，这可以通过给移动用户分配一定比例的缓存空间来让用户直接将内容上传。

CDN 需要考虑的另外一个参数是移动用户的导航行为，影响这个参数的最显著因素是用户实际使用的移动设备。然而，移动设备输入能力有限，例如手机键盘相对于计算机键盘而言，允许输入的文本很少。移动用户也必须接受较慢同时却费用昂贵的下载服务。所以，这些特点导致了用户采用移动互联网和有线网络去获取信息所带来的差别。

用户的移动性促进了移动 CDN 中面向服务的手机定位技术的发展。假设一个移动用户使用了支持 CDN 的应用，当他在移动时应该确保这个应用能够被其他的代理缓存服务器复制，以便该应用能够始终“伴随”用户。手机定位技术甚至还可以用来检测用户的互联网连接速度，这一点对于网站的拥有者非常重要，因为他们希望可以向其客户证明多媒体应用（如广告）是可行的。

此外，典型的 CDN 不提供任何涉及用户（即与 CDN 交互的用户）实时状态和底层网络设施的监控机制。然而，考虑到移动无线网络设施自身存在着若干局限性，这些监控机制是移动 CDN 中支持内容发布的关键组成部分，这些局限性简要介绍如下。

（1）网络频繁断线

用户的随机性或组织移动性会严重影响网络的连通性，主要表现在：①与基站或者其他邻近移动设备的连接时间短；②高楼、树木、车辆等障碍物都会显著影响无线信号的质量；③用户临时处于无线网覆盖范围外的几率。以上因素都会造成带宽下降或彻底失去连接，最终导致信息丢失。

（2）网络分片

上述频繁断网的情况会导致移动无线网络分片，使移动站点之间的端对端连接失效，从而降低了信息的有效性。

（3）移动节点约束

大多数的移动设备会面对很多约束，如处理能力、存储容量和最重要的续航时间等。这些限制是源于设备通常是用电池供电，而且其小尺寸也就意味着处理能力和存储容量不会太好。

鉴于移动无线网络的以上限制，CDN 很有必要了解移动用户的状态，这样可以减少整个网络的流量。例如，一个移动用户要将一个播客（podcast）下载到 MP3 播放器，但是在下载的过程中，一旦电池的电量耗尽则会导致设备掉线。在这种情况下，CDN 应该能获知用户当前的状态，并停止传输数据。

以上这些论述可以总结为：以低成本的形式有效地为移动用户发布信息非常具有挑战性，尤其在应用服务（如流媒体、动态内容）日益增长及移动无线环境本身存在局限的情况下。移动 CDN 的设施将面对这些挑战，下一节将深入介绍移动 CDN。

14.3 移动 CDN

与有线 CDN 不同，移动 CDN 通过配置一系列的无线网络（如蜂窝网、Wi - Fi 等）来向移动设备提供诸如动态数据和丰富的多媒体内容传输等高质量的服务。特别地，移动 CDN 的网络设施可以分解为有线网络设施和无线网络设施。前者在有线 CDN 环境下负责提供源服务器与代理缓存服务器之间的连接以及代理缓存服务器与网络组件之间的连接，如交换机、路由器、3G/GSM 基站（BS）、Wi - Fi 无线接入点（AP）。无线网络设施负责无线 CDN 环境下静态用户和移动用户的通信和内容发布。所以，客户端—服务器之间的通信被三个通信流所替代：①用户与无线网络边缘（BS 或 AP），②无线网络边缘（BS 或 AP）与代理缓存服务器，③代理缓存服务器与源服务器。一个典型的移动 CDN 如图 14.1 所示。

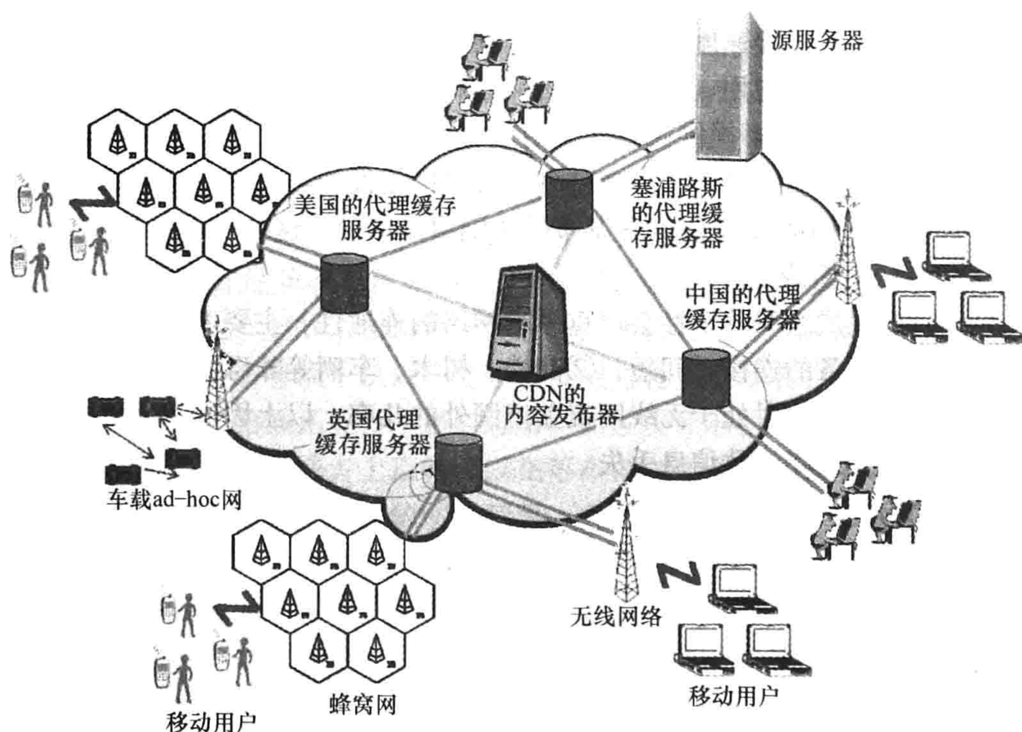


图 14.1 一个典型的移动 CDN

考虑到代理缓存服务器应该被安装在无线网络的 3G/GSM 基站（BS）及 Wi - Fi 接入点（AP）“附近”，所以移动 CDN 的拓扑结构应该重新配置，并重新设计代理缓存服务器的放置地点，以充分覆盖移动无线设施。由于基站数量巨大，移动 CDN 应该比典型的传统 CDN 集成更多的代理缓存服务器。

鉴于代理缓存服务器的架构，它们的缓存应该被分成两个部分来支持面向用户的服务。一部分用来复制（与 CDN 运营商有合约的）源服务器的待发布内容，另外一部分用来存储用户的上传内容。进一步而言，一个移动 CDN 的代理缓存服务器也应

该向移动用户提供地理位置的定位服务。这样的服务需要定义一个服务探索协议，来通过移动用户的位置（移动用户此刻的地理区域或具体位置）选择代理缓存服务器，这样便于将应用转到距离移动用户最近的代理缓存服务器。

除了有关移动 CDN 的架构搭建问题，被复制内容的发布策略也十分重要。大多数 CDN 提供商使用非协作式或协作式的拉取方法[27]。这两种方法的主要特点是它们都是反应式的；一个数据对象只有在用户请求的情况下才会被缓存，所以当用户数量很多时这些方法将导致通信成本（信息交互量）的陡然增加。另外，当内容变化非常快或同步需求十分严格时，这些方法将不能保证可靠性。鉴于上述局限，这种拉取式方法在移动无线环境下是不可行的，而是应该着重研究基于协作的内容推送方法。与拉取方法（非协作式方法和协作式方法）需要等待用户请求信息不同，协作式推送方法让源服务器预先将信息推送至缓存服务器以减少访问耗时。Chen 等人[9]的相关研究显示，此种方法与其他方法相比是最优的，内容将预先从源服务器推送至代理缓存服务器。当收到用户请求时，若代理缓存服务器有目标副本，则可以迅速响应该用户请求；若代理缓存服务器没有目标副本，其转向拥有该副本的最近的代理缓存服务器来响应用户请求。若被请求的目标文件在所有代理缓存服务器中都没有副本，则转向源服务器。这种方法要求的代理缓存服务器间的协作会带来额外的通信成本和管理成本，但这些成本也被代理缓存服务器间带宽共享和减少复制冗余所带来的优势抵消了。同时，这种方法也可以减少维护缓存一致性的成本。

表 14.1 列举了典型 CDN 和移动 CDN 的主要区别。

表 14.1　典型 CDN 和移动 CDN 的对比

特性	典型 CDN	移动 CDN
内容类型	静态、动态流	静态、动态流
用户位置	固定的	移动的
代理缓存服务器定位	固定的	固定的
代理缓存服务器拓扑	靠近互联网服务提供商	靠近基站
副本维护费用	中	高
服务	应用服务	面向地理定位的应用服务；面向用户的服务
内容外包策略	协作/非协作拉取法	协作推送法

综上所述，一个移动 CDN 应该由以下几个部分组成：

1）分布于世界各地的一系列代理缓存服务器，用来缓存源服务器内容。代理缓存服务器是非移动性的，而且位于移动基站附近。

2）一个网络设施（有线和无线），负责将内容请求转到最优地点和最优的代理缓存服务器。

3）监测网络设施的机制，负责实时监测有效带宽、反应时间和其他资源拥塞

情况。

4）缓存管理器，用来有效管理上传到代理缓存服务器的内容。

5）内容位置管理器，负责管理内容的位置和数据预取调度。

6）记账机制，向源服务器和 CDN 提供商提供日志和相关信息。

以上各个部分互相协作来将目标内容分发至移动用户。在这里，内容指一个随时间变化的目标集。例如，内容可以是一个特定的网站、一个流服务、一个网页服务或任何分布式应用的集合。当移动用户需要请求某个目标内容时，用户的请求被立刻送至在地理意义上最邻近的代理缓存服务器。如果这个代理缓存服务器恰好存有这个目标内容，请求会立刻得到响应。否则，移动 CDN 将转向拥有该目标副本的最近的代理缓存服务器来响应用户请求。

针对用户请求的高可变性和内容动态变化的特性，移动 CDN 必须将内容管理策略整合到代理缓存服务器的缓存中。如何选择外包内容[40]和如何实施外包[19, 28, 44, 51]有不同的方法。缓存管理器负责有选择地将内容复制到代理缓存服务器，并且保持内容的更新。特别地，需要注意缓存一致性的问题，详细内容可参见参考文献［30］。所以，为了应对用户请求的临时变化，缓存管理器需进行周期性检查。另外，内容位置管理器负责将外包的内容复制到代理缓存服务器。最后还需要一个可以实时监控用户状态的机制，因为用户很有可能由于处于移动状态而不能接收到其请求的内容。

考虑到 CDN 的市场情况，尽管在市面上已经有了一些 CDN 提供商㊀，但是移动 CDN 提供商还是很少。据本章作者所知，Ortiva Wireless㊁是目前仅有的一家移动 CDN 提供商，能够在高变化度的无线网络覆盖条件下将视频传递给移动用户。通过对移动电视、视频、音频、网页内容的分发进行优化，Ortiva 可以在任何无线网络间高质量地传输数据，在没有对网络基础设施进行大幅度更改的情况下，Ortiva 实现了业务、覆盖面和网络容量的拓展。

14.4 移动 CDN 的无线网络设施

如前所述，移动 CDN 的网络设施可以分解为无线设施和有线设施。有线设施提供 CDN 所要求的适应性和容错性，CDN 的一大部分位于互联网的主干网处，需要进行高冗余度的配置。这一节重点描述现有的两种无线网络设施[37]，并讨论它们在移动 CDN 中的适用性。

14.4.1 在集中式无线网络设施下的移动 CDN

蜂窝网络和 Wi－Fi 网络是两个典型的集中式无线网络设施。从图 14.2 可以看到，所有的用户均与一个中心进行通信。在蜂窝网络内，中心通常是一个 3G 或 GSM 的基站（BS），在 Wi－Fi 网络中则是 IEEE 802.11 接入点（AP）[37]。无论是基站还是接入点，

㊀ 现有 CDN 提供商的完整列表参见参考文献［29］。——原书注

㊁ http：//www．ortivawireless．com/。——原书注

都作为无线用户的硬件桥梁来负责无线通道的分配和控制。另外，基站/接入点持续跟踪用户，这样可以当用户不在无线覆盖范围内或下线时停止内容的分发。

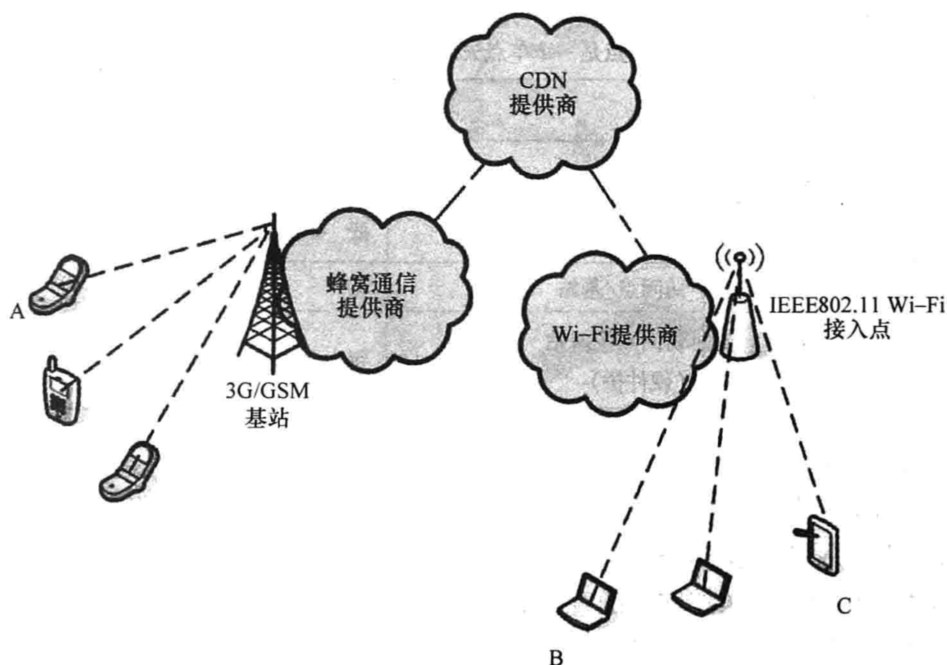


图 14.2　集中式无线网络设施

集中式的无线网络设施为移动 CDN 提供了一个很好的架构。例如，图 14.2 中的用户 A 拥有一部内置摄像头的手机（已入网登记）。A 目前正在度假，他/她拍了一个观光视频，并想将这个视频上传至在线博客，这样 Wi - Fi 网络的用户 B 和 C 就可以访问该视频了。只要用户 A 在蜂窝网络的覆盖范围内，就可以通过手机将视频传输到这个区域的基站上。接下来，基站将这个视频通过蜂窝网络发送到博客的托管服务器上。移动 CDN 提供商就负责将这个视频推送至离用户 B 和 C 尽可能近的 Wi - Fi 网络中。

但是，这种集中式无线网络也存在几个问题。负责协调无线环境下通信的集中式管理机构限制了移动环境中的可扩展性和适应性[10]。所以，为了能够满足通信覆盖和提供服务质量（QoS），在配置时需要慎重对待，并通过系统化的方式进行。

14.4.2　在 Ad - Hoc 无线网络设施下的移动 CDN

与集中式网络设施相比，Ad - Hoc 无线网络设施在移动环境下具有几个优点。从技术的角度讲，Ad - Hoc 无线网络由自治节点组成，节点间通过无线点对点连接进行内部通信。通信通过多跳方式进行，不依赖于任何一个中心机构来负责信道分配、拓扑信息或者内容分发。与集中式网络相比，在用户数量增加时 Ad - Hoc 网络的扩展可以采用一个经济上划算的解决方案[38, 50]。这两种网络类型的主要区别参见表 14.2。

表 14.2 集中式无线设施与 Ad – Hoc 无线网络设施对比表

特　　性	集中式无线设施	Ad – Hoc 无线设施
容错能力	无 （访问点是一个单点失效）	有
节点数增加时的可扩展性	无	有
节点的自组织和自配置	无	有
维护和扩张成本	高；需要额外访问点	低
中央控制机构	访问点/基站	无
桥接	访问点或基站 （硬件桥）	每个节点都充当其他节点的桥（软件桥）

移动 Ad – Hoc 网络（MANET）是一个以点对点和 P2P 方式建立起来的无线网络，具有动态和任意的拓扑结构，实现信息检索和分发。MANET 的基本特性如下所示：

（1） Ad – Hoc 和 P_2P 网络的连接性

这样的网络具有自组织和自配置的能力。Ad – Hoc 设施是非集中式的，不依赖于任何固定设施。非中心化的架构突出了网络的容错性，不会出现单点失效。另外，Ad – Hoc的连接性保证了网络的可扩展性，因为没有必要再去建立额外的网络设施。它们的这种连接特性是由设施的 Ad – Hoc 连接本质决定的，任何节点在网络中的地位都是平等的，在任何时候一个节点既可以是主机也可以是路由器[21]。

（2） 网络拓扑

动态性来自节点组织的不可预知性和随机性。更进一步说，这是由于移动节点在给定的任何时间点能够加入或离开无线网络，而无须预先通知网络。固定节点可以与移动节点建立对等，给其他固定设施或互联网提供网关功能。

（3） 移动节点约束

移动节点依赖于充电电池的供电，这就意味着其续航时间是有限的。另外，它们的处理和存储能力也由于它们的尺寸相对较小而受限。

由于在快速变化、非确定性的移动环境下移动 Ad – Hoc 网络的设施对于信息发布有卓越的性能[15, 42]，因而 MANET 引起了研究人员和业界的重视。如图 14.3 所示，用户 A 希望在一个多媒体在线网店（在移动 CDN 上已经注册入网）下载一个视频到其笔记本电脑。但是，由于他/她此时正处于 Wi – Fi 或蜂窝网络提供商的无线覆盖范围外，所以他/她转换到 Ad – Hoc 模式，加入到由用户 B、C、D、F 组成的 MANET 中㊀。注意到用户 E 也是 MANET 网络中的一部分，并且可以使网络中的其他成员连接到互联网，所以用户 A 的请求就通过 MANET 先传递到用户 E，再从 E 传递至多媒体在线网店。然后，相应的移动 CDN 会受理请求并将请求转至与用户 E 最邻近的代理缓存服务器上。为了补偿移动节点的局限性，上述移动 CDN 会以特定的编码及压缩方式提供这个视频[2]。

㊀ 安全问题及用户之间的信任问题在无线 Ad – Hoc 网络中非常重要，但这不是本章的内容。——原书注

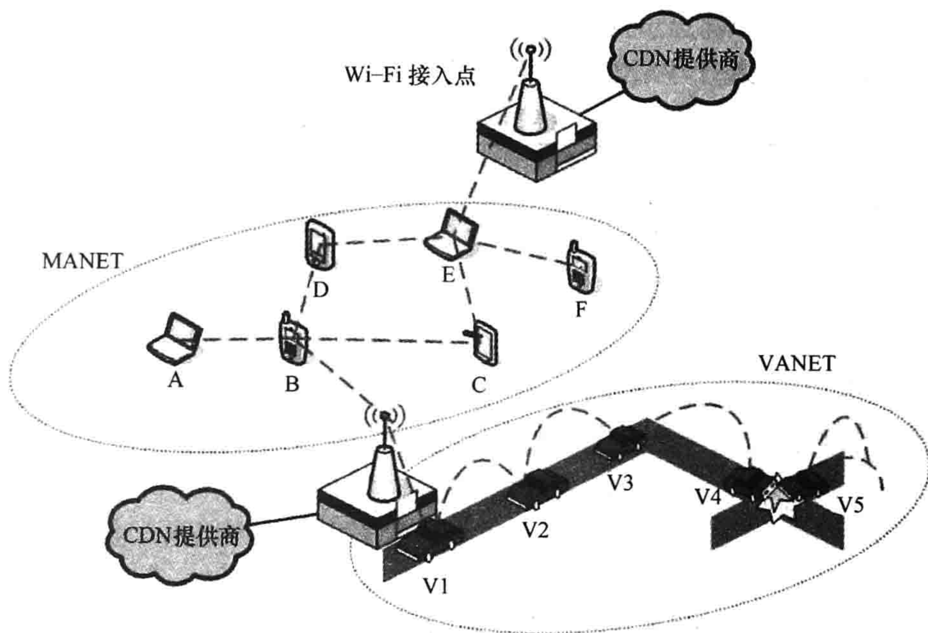


图 14.3　一个典型的 Ad - Hoc 无线网络设施

MANET 的一个特例是车辆 Ad - Hoc 网络（VANET），是由受道路约束的车辆和固定在路边的提供位置服务和互联网网关功能的节点组成。特别地，VANET 是为了实现智能交通系统而开发的一个相应的设施[2]。VANET 与 MANET 有以下不同：

（1）网络拓扑

由于车辆的移动性更强，VANET 比 MANET 更动态化。但是，VANET 中车辆节点的移动可以预见，因为它们通常都在特定的道路范围内，以相对规范的方式行驶。与 MANET 一样，路边节点与车辆节点进行对等互联，并作为固定设施和互联网的网关。同时，路边节点可向车辆节点广播位置信息。

（2）节点约束

VANET 没有电源和续航时间的约束，因为车辆节点由车载电源持续供电。

下面将介绍将移动 CDN 技术应用到 VANET 中的好处。例如，车辆 V_1，…，V_5 加入了一个 VANET，假设 V_4 与 V_5 在路口处发生交通事故，V_4 和 V_5 立刻向网络的其他成员广播有关事故的照片或录像信息。该警示信息被标记为高优先级，并在 VANET 中进行传播。当 V_1 收到警示信息后，它利用附近的互联网基站将这起交通事故传送至在线交通监控系统。移动 CDN 会将这个配有照片和录像的警示信息传送至代理缓存服务器。这使得应急事件小组可以在最短的时间内对事故有全面的了解。同时，当其他正驶向事故地点的车辆通过车辆监控系统看到这个警示信息后，可以采取改变路线等措施来避免拥堵。一般而言，移动 CDN 在交通环境中的作用包括：

1）通过移动互联网向车辆提供动态内容和流内容。

2）向有关车辆和机构发布道路交通信息，如拥堵、交通事故和临时绕行等信息。

3）定位服务，如各种场所的位置信息（如宾馆、饭店、加油站、停车场等）。

4）将车辆传感器收集到的环境测量信息进行发布。

5）支持分布式游戏。

最后，表 14.3 列举了 VANET 与 MANET 的主要区别。

表 14.3 MANET 与 VANET 对比表

特　性	MANET	VANET
移动节点	笔记本计算机、智能手机、PDA	车辆
节点移动性	随机 - 不可预测	有组织的 - 可预测的（在道路约束下）
节点约束性	有限（和电池约束相关）	不存在电源不足问题
移动性	高	十分高
网络拓扑	动态	动态
支持的网络技术	UMTS，GSM，Wi - Fi，Bluetooth	DSRC，UMTS，GSM，Wi - Fi

14.5 写给使用者

现有文献中已经针对学术界和业界中的若干项目进行了研究，具体内容是在源服务器与移动节点间开发并部署中间组件，以便进行两者之间的无缝信息发布[12]。这些中间组件配置在无线设施中，以优化网络连接和带宽利用率。这一节将介绍移动环境中使用中间组件的有关方法，以及如何将这些方法应用到移动 CDN 中。

IBM 的 Web Express 系统采用了客户端—拦截—服务器的无线计算模型[18]，对资源有限的移动节点的网页访问进行优化。这个模型利用优化通信的拦截技术来改善无线连接的响应时间。具体来说，Web Express 系统有两个关键组件：配置在移动节点的客户端拦截（CSI）组件和配置无线网络的服务器端拦截（SSI）组件。CSI 截获移动节点的请求，并与 SSI 进行协作来减少数据传输量，改善内容在无线连接中的可用性。Web Express 系统采用了若干种优化方法来解决在客户端 CSI 与服务器端 SSI 缓存、计算已有缓存内容与新响应间差别、在单一会话下的请求复用以及缩小 HTTP 头等问题。

作者认为，在移动内容网络的应用中，Web Express 系统或更特定的客户端/拦截模型可以采用将 CSI 从源服务器移至代理缓存服务器的方法。重定位可以进一步地减小移动用户通过无线网络的网页访问时长，通过上述的优化方法也可以减轻网络拥堵。

Dikaiakos 等人[13]提出了一个独立的信息传输协议，在 VANET 中的车辆和路边固定节点间传播信息。车辆信息传输协议（VITP）是一个属于应用层的无状态协议，它规定了车辆节点间通信的语法及语义，但同时又隶属于 VANET 协议。他们提出的方法将 VITP 配置到车辆上来支持基于位置和面向交通的服务。也就是说，查询请求被传输到一个特定的地理区域来检索所需的内容。在这个目的地（目标）位置内，一个名为虚拟 Ad - Hoc 服务器（VAHS）的动态集被立刻建立，用来计算对位置查询的响应。当返回条件满足时，VAHS 中的节点彼此协作，发送一个对 VITP 请求的响应。VITP 查询按照请求—响应的原则进行。另外，协议通过缓存控制头（包含在 VITP 查询请求中）进行数据缓存。如果在一个对等节点的缓存服务器中存在一个满足 VITP 查询的内容，这个节点就可以使用上述头来决定是使用本地缓存进行响应还是向目标位置重传查询命令。

在移动 CDN 中，路边固定的 VITP 节点可以作为代理缓存服务器使用。与车辆相

比，这样的节点有更强的处理和存储能力、更大的无线电波覆盖范围和连续的高带宽互联网连接。每一个路边节点可以被指定为负责该区域的代理缓存服务器，因此这些代理缓存服务器可以缓存频繁的 VITP 信息和位置信息。

另外，移动 CDN 可以向现有车辆应用提供服务，如 CarNet[24]、TrafficView [26]等。特别地，CarNet 是最早应用内容分发技术来提供车辆服务的应用之一。另外，TrafficView 也是一个典型的 CDN 框架，用来进行交通信息、路况信息、路线规划信息和应急信息等内容的发布。TrafficView 可以显示实时动态的交通状况，以作为传统的车载 GPS 的有效补充。

最后，Daly 和 Haahr[11] 提出使用运行有 SimBet 路由算法的中间组件来对 MANET 中的信息发布进行改进。SimBet 使用一个基于社会网络分析的指标来确定在网络中移动节点的中心。为了找到中心节点，SimBet 使用一个称为中介中心性（betweenness centrality）的概念。所以，如果发送节点不知道目标节点，信息就会被路由到中心节点。对于移动 CDN，这个方法可以被应用在某个节点上，这样就可以将被请求的内容发送到无线网络中的中心节点；当用户请求的内容正好暂时不在覆盖范围之内（或掉线）时，这个方法就可以派上用场。

14.6　实施和实验

在实际设计和实施一项新技术之前，首先要对实际情况下的策略和应用、实现、CDN 设施进行一系列仿真实验。之所以要进行仿真，是因为在真实环境中进行实时的实验非常困难。从实现角度来看，以下几种仿真方法可以用来评估一个移动 CDN 设施的性能：

（1）仿真测试平台

大规模的仿真测试平台为新的 CDN 技术、策略、应用的实现和评估提供了一个必要的设施。如 PlanetLab㊀就是一个节点遍布全球各地的全球性覆盖网。事实上，PlanetLab 可作为一个 CDN 测试平台，用于配置一个全球性的服务和跨越全球范围的应用。这个 CDN 测试平台由高性能的代理缓存服务器网组成，在许多 PlanetLab 节点上都配置了这样的代理缓存服务器。在 PlanetLab 之上已经建有两个学术性 CDN：CoDeeN㊁和 Coral㊂。学术性 CDN 在第 1 章已经给出了详细的介绍。

（2）仿真软件

一般来说，在所有的科学研究领域里，仿真软件都是产品设计和测试的必备工具。CDN 仿真器也毫无例外地成为 CDN 研究和业界中的必备品。这些工具为模拟 CDN 应用和设施提供了手段，但是没有任何具体支出，也不需进行硬件的安装和配置，仿真结果是可复现的，也便于分析，因为仿真环境不存在不可预料和不可控的影

㊀ http://www.planet-lab.org。——原书注

㊁ http://codeen.cs.princeton.edu。——原书注

㊂ http://www.coralcdn.org。——原书注

响因素（如不希望的外部流量）。相反，在实际网络设施中，研究人员则可能会遇到上述影响因素。CDN 仿真器模拟一系列的计算机行为，也就是说，代表一台源服务器向用户可靠和高效地分发内容。本书第 5 章介绍了一个仿真分析工具 CDNsim。

但是，上述的 CDN 仿真软件和测试平台只考虑了固定、有线的网络设施，并不支持移动 CDN 的仿真和评估，因为在这些软件中没有考虑到用户的移动性、无线传输媒介和其他在 14.3 节中提到的移动特性。作者认为，这些 CDN 仿真软件或测试平台应该通过开发新的可扩展的模块来支持移动 CDN。这些模块应能够提供：

（1）真实的移动性轨迹

为了实现移动节点的移动行为模拟，可以通过常见的移动模型来产生一系列的移动轨迹。例如，一个支持产生移动轨迹的模块可以集成 SUMO 的相关功能[20]。SUMO 是一个知名的微型便携式交通模拟器，它可以模拟城市环境下车辆行为的移动轨迹。另外，相关模块的研究人员可以考虑 CosMos，这是一个由 Günes 和 Siekermann[16] 提出的通信和移动情景模拟器。与其他基于单一移动模型的移动轨迹产生器不同，CosMos 集成了几个移动性模型，所以它可以为无线网络仿真提供更真实的移动模式。

（2）无线环境的支持

如本章前面所提到的，无线环境与有线环境呈现着不同的特性。因此，模拟无线环境下的特性对于移动 CDN 的仿真和评估至关重要。支持无线环境的新模块应该能够准确地模拟设施和 Ad - Hoc 环境下的振荡信号强度、带宽、间歇性、连接性。甚至，这种模型可以进一步用于模拟波的传播特性，如多径、衰减和衍射特性。

（3）移动资源受限的支持

移动 CDN 模拟器应该考虑移动节点的一个重要的特点，即可用资源的有限性。具体地说，组成移动 CDN 的大多数节点都受到功率限制。所以，应该开发在不同的情景下（睡眠、发送、接收、空闲等状态）节点的能耗模型，并在新的模块中实现。这样的模型最终将用来正确地模拟在上述限制条件下移动节点的行为，并允许用来试验各种节能技术。

ORBIT⊖是一个支持无线技术研究的无线网络测试平台，由若干大规模无线网络组成，网络中含有 400 个支持高速蜂窝（3G）和 IEEE 802.11 协议的节点。与其他大型测试平台相比，ORBIT 的优点在于支持移动仿真，而移动性是通过布朗运动或随机路径点模型得到的，这就具备了仿真各种无线应用和无线技术（如 MANET 和面向位置服务）的能力。

从试验的角度看，我们希望无线 CDN 可以减少用户的平均响应时间，增加代理缓存服务器缓存字节命中率。一方面，平均响应时间代表用户等待请求响应的时间，另一方面，字节命中率代表着网络的性能。另外，我们期望移动 CDN 中的代理缓存服务器可以借助彼此之间高度的协作性而具备较低的复制冗余度。这一点非常重要，因为高数据冗余度会浪费 CDN 提供商的资金[27]。最后，我们期望移动 CDN 能够通过

⊖ 下一代无线网络开放访问研究测试平台（Open - Access Research Testbed for Next - Generation Wireless Networks），参见 http：//www. orbit - lab. org。——原书注

响应大多数用户请求来改善对移动用户的服务质量。如果没有这样一个设施，许多移动用户将面临服务被拒绝的情况。

14.7　未来研究方向

移动无线网络的特点是用户需求的多变性，以及动态内容和多媒体应用的高需求性。例如，一个司机就希望能在旅程中收到任何关于紧急情况的多媒体信息。移动无线网络的另一个特点是资源的缺乏性，例如某用户希望将一段视频上传至服务器以腾出存储空间拍摄更多的照片或录像。上述问题必须得到解决，接下来将探讨移动 CDN 未来的研究方向：内容放置技术、动态内容和移动流媒体的发布技术。

14.7.1　内容放置技术

内容放置技术解决的是内容被复制到何处的问题，可以通过一些基于请求和资源约束的目标函数进行优化设计。对静态用户需求而言，这个问题已经得到了详细的研究[19,28,44,51]。但是，由于移动性和资源的约束，现有的方案并不适用于移动无线环境。所以，必须要研究针对于动态内容、多媒体应用和用户移动性的新方法。

针对动态用户需求的内容放置方法是关注的重点[8,25,31,33]。Chen 等人[8]提出了一种动态放置副本的算法，该算法利用有限的网络拓扑知识将副本动态地放置和组织为一个应用层的组播树，旨在兼顾用户等待时间和服务器容量。在参考文献［25］中，作者提出另外一种动态内容放置框架，将动态内容放置的优化问题描述为一个半马尔可夫决策过程，用户的请求被假定服从马尔可夫模型。Presti 等人[31]利用非线性整数规划方程来解决动态内容放置的问题。Rabinovich 等人[33]使用应用 CDN（Application CDN，ACDN）来进行动态内容分发，提出了一个基于用户历史需求的启发式方法。

但是，以上的内容放置技术都没有在无线网络设施上进行过评估。鉴于此，参考文献［1］针对移动 CDN 提出了在线启发式动态内容放置算法（称为在线 MDCDN），该算法是基于二次指数平滑（DES）的统计预测方法，通过用户需求的变化来预测每一个代理缓存服务器的未来需求。这些预测被用来决定是否需要添加一个新的内容副本或者需要移除一个已有的内容副本，以最小化网络主干的总体流量。为了进行验证，作者使用了一个移动仿真器[22]和一个 20km 半径的城市模型，城市模型根据人口密度和自然约束（如河流、公路等）被划分为若干个区域。

Miranda 等人[23]提出了一个称为 DbC 的分布式算法，将内容尽可能均匀地放置到无线 Ad－Hoc 网络中的所有服务器上。这样，数据的副本之间都放置的足够远来防止出现过度冗余。另外，副本对于每一个终端用户又相距足够近，这就改善了信息传播。仿真结果表明，DbC 改善了整个网络的数据传播。另外，在参考文献［17］中提出了三种有效的无线对等网络设施内容放置技术，这些方法将副本放置在移动主机上，分别考虑了对象的PT 值[⊖]（扩展静态访问频率，E－SAF）、对象与相邻移动主机

⊖ PT 值被定义为距下次对象更新的所剩时间与对象流行度的乘积。——原书注

的 PT 值（扩展动态访问频率和邻近度，E-DAFN）、或者对象与整个网络拓扑的 PT 值（基于扩展动态连接性的分组，E-DCG）。参考文献［17］中的实验表明，E-DCG 具有最高的访问能力，而 E-SAF 方法的流量最低。

14.7.2 动态内容的分发

分发动态内容的应用对延时非常敏感。例如，一个司机想知道走哪条路才可以避免交通拥堵，在这种情况，哪怕是几秒钟的延时都是不能容忍的。所以，移动环境下动态内容的有效分发具有相当重要的意义。为了解决这个问题，研究人员提出了很多方法和技术来加速动态内容的分发[1,5,34]，这些方法在某些商业系统（如 IBM 的 Websphere 边缘服务㊀和 Akamai 的 EdgeSuite 网络㊁）中的应用已经证明了动态内容分发技术的重要性和有效性。

近些年基于片段的策略引起了学术界的广泛重视[5,34]。实验结果显示，用片段组成网页比从头构建一个网页具有更低的开销[5]。Akamai 也使用了 ESI（Edge Side Includes）技术来增强基于片段的策略。Ramaswamy 等人[34]提出了一个在动态产生的网页中能自动检测和标记感兴趣片段（即低成本的缓存单元）的新方法，其要点是如果一个片段被多个网页分享或它受关注的时间长，或本身具有个性化的特点，那么它就会被当做感兴趣的片段。

还有一些与缓存网页片段不同的动态内容分发方法，它们是在代理缓存服务器上对产生网页的手段进行缓存。之所以采取这个方法，是因为按需产生内容的时间比单纯读取任何动态内容所用的时间长，因为前者需要多次查询数据库。所以，一个简单的想法就是在 CDN 代理缓存服务器上缓存应用代码，并且保持数据中心化。这个技术是 Akamai 和 ACDN[33]中边缘计算方法的基本模块。这种方法的一个缺点是所有的用户请求都必须在一个中心数据库中得到响应，因此这种中心化的架构会导致性能瓶颈。为了解决这个问题，一种方法是在每一个代理缓存服务器上建立一个数据库的部分副本，实现该方法的系统被称为内容已知缓存（CAC）系统；另一种方法是对于应用代码产生的数据库查询结果进行缓存，该方法被称为内容未知缓存（CBC）[35]。每当收到一个查询时，就会检查查询的结果是否被缓存。Sivasubramanian 等人[41]的实验结果显示，当查询负载呈现高度局部性时，CBC 具有很好的性能；相反，如果查询负载呈现较低的局部性，则 CAC 的性能更好些。

14.7.3 移动流媒体的分发

随着下一代移动系统带宽的增加，流媒体成为了内容分发服务中的重要组成部分。所以，在移动环境中有效地发布流媒体是个非常有挑战的课题。通常，流媒体内容可以分为如下几种：

㊀ IBM WebSphere 应用服务器：http：//www-306. ibm. com/software/webservers/appserv/was/。——原书注

㊁ http：//www. akamai. com。——原书注

（1）直播内容

内容从编码器立即发布到多媒体服务器，然后到终端用户。

（2）点播内容

内容在编码器完成编码后，先以流媒体文件的形式存储在媒体服务器上，随后供终端用户下载。

媒体服务器是一个专用的、运行在通用服务器上的一个软件，负责响应用户对经编码的数字化内容的请求。当用户对某个特定内容发出请求时，媒体服务器以相应的视频或音频流来响应。Roy 等人[36]讨论了这种服务器在 CDN 中的设计要求。但是，由于流媒体的严格需求、无线网络的局限性和用户的移动性，向大量移动用户发布多媒体内容时会产生一些严重的问题。

移动 CDN 可以通过将移动用户的大量流媒体需求分配至代理缓存服务器来解决上面所提到的问题。甚至，CDN 代理缓存服务器可以采用最新的压缩技术（缓存、编码等）来改善流媒体的发布。具体地说，一个移动流媒体 CDN 应该考虑以下问题：

（1）代理缓存服务器的选择

当用户请求某个目标时，他/她的请求应该被代理缓存服务器直接响应。为了实现上述目标，Yoshimura 等人[49]提出了一个移动流媒体内容发布的网络结构，其中代理缓存服务器的选择是由一个 SMIL㊀文件决定的。包含有代理缓存服务器状态信息的 SMIL 文件存储在流媒体服务器上，移动用户可以通过读 SMIL 文件来选择 CDN 中可以对其提供最好响应的代理缓存服务器。

（2）媒体缓存

早期的研究表明，将缓存技术整合到 CDN 中可以提高 CDN 的性能。问题是如何决定什么样的媒体流应该被缓存在代理缓存服务器的磁盘上。一个解决办法是存储所有的多媒体流，但这个办法并不可行，因为多媒体流需要的存储空间相当大。所以，应该在移动多媒体流 CDN 中使用有效的数据管理方法，这样缓存管理器就可以借助这些方法有效地将内容复制到每个代理缓存服务器。一个简单的办法就是将对象分段，也就是说将多媒体对象分成更小的单元去缓存，这样可以提高存储效率和网络利用率。在现有文献中有一些方法用于缓存分段[7]，如前缀缓存[39]或变尺寸分段[47]都是有代表性的缓存分段技术。

（3）管理会话的切换

在流媒体中应用中，用户会话具有持久性。但是，在移动环境中这种持久性会产生一个问题：如何以有效的方式对代理缓存服务器间的会话切换（handoff）进行管理。在切换时应保证没有内容丢失，且流媒体数据对于视频播放器仍保持尽可能的流畅。在参考文献［49］中，使用 SMIL 文件用来控制会话切换。

Apostolopoulos 等人[3]提出了一个移动流媒体 CDN（MSM－CDN）框架，这个框架

㊀ 同步多媒体集成语言（Synchronized Multimedia Integration Language，SML）是一个 W3C 推荐的 XML 标记语言，用于描述多媒体的特征。SMIL 定义了对于时间、布局、动画、可视化过渡以及媒体嵌入等的标识。——原书注

允许在下一代移动网络中进行媒体分发。这个移动流媒体 CDN 框架采用模块化设计，支持与其他系统和底层网络之间的操作。其中，用来缓存媒体流的覆盖服务器是移动流媒体 CDN 的基本组成部分（这里，一个覆盖服务器可被视为代理缓存服务器）。媒体的分发可以通过流接口和数据传输接口进行，在移动流媒体 CDN 中的接口方便了媒体流至移动用户的分发。在参考文献［49］中提出了另外一种移动流媒体 CDN 框架，其创新之处在于让所有与 CDN 有关的技术都可以在经 SMIL 修改后得到应用。

14.8 结论

移动内容网络的新技术（如 GSM/3G、Wi - Fi 等）使无线网络设施可以支持那些带宽密集型服务，如流媒体、移动电视等。通常被移动用户请求的那些信息，要么是动态内容，要么是媒体应用。考虑到移动用户对信息的需求越来越大，我们迫切需要新技术的诞生，以改善信息分发的效率。但是，典型的 CDN 已经不能满足移动无线网络中的这些需求，因为这种设施存在固有的局限性，无法满足移动设备的需要。也就是说没有考虑到用户的移动性。针对这种状况，移动 CDN 可以通过加速在无线网络设施中的信息发布来解决这些问题。移动 CDN 与典型 CDN 在代理缓存服务器的拓扑结构、内容外包策略和应用服务上都有区别。

本章概述了移动 CDN 在移动网络设施的最新发展过程中所扮演的角色，同时讨论了如何通过新兴的移动 CDN 技术来改善信息分发，为此我们列出了移动 CDN 的主要特点。随后，我们提出了在移动 CDN 中常见的动态内容分发技术、内容放置技术和移动流媒体技术。最后，我们讨论了移动 CDN 的网络设施，并探讨了现有的信息分发技术。

总的来说，移动无线环境下的信息分发是一项有意义、有挑战的研究内容。移动 CDN 的出现，为业界和学术界开辟了新的视角。尽管在某些方面已经有了若干进展，但无论在理论上还是在实际应用上都还存在着很大的发展空间，在移动计算和移动网络技术领域内需要继续进行新的探索。

参考文献

[1] Aioffi, W.M., Mateus, G.R., Almeida, J.M., Loureiro, A.A.F.: Dynamic content distribution for mobile enterprise networks. IEEE Journal on Selected Areas on Communication **23**(10) (2005)

[2] Anda, J., LeBrum, J., Ghosal, D., Chuah, C.N., Zhang, M.: Vgrid: Vehicular ad-hoc networking and computing grid for intelligent traffic control. In: Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st, Vol. 5, pp. 2905–2909 (2005)

[3] Apostolopoulos, J.G., Wee, S., tian Tan, W.: Performance of a multiple description streaming media content delivery network. In: Proceedings of the 2002 International Conference on Image Processing (ICIP 2002), pp. 189–192. Rochester, New York, USA (2002)

[4] Chae, M., Kim, J.: What's so different about the mobile internet? Commun. ACM **46**(12), 240–247 (2003)

[5] Challenger, J., Dantzig, P., Iyengar, A., Witting, K.: A fragment-based approach for efficiently creating dynamic web content. ACM Trans. Inter. Tech. **5**(2), 359–389 (2005)

[6] Chand, N., Joshi, R.C., Misra, M.: Cooperative caching in mobile ad hoc networks based on data utility. Mobile Information Systems **3**(1), 19–37 (2007)
[7] Chen, S., Wang, H., Zhang, X., Shen, B., Wee, S.: Segment-based proxy caching for internet streaming media delivery. IEEE MultiMedia **12**(3), 59–67 (2005)
[8] Chen, Y., Katz, R.H., Kubiatowicz, J.: Dynamic replica placement for scalable content delivery. In: Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS), pp. 306–318. Cambridge, USA (2002)
[9] Chen, Y., Qiu, L., Chen, W., Nguyen, L., Katz, R.: Efficient and adaptive web replication using content clustering. IEEE Journal on Selected Areas in Communications **21**(6), 979–994 (2003)
[10] Chisalita, I., Shahmehri, N.: A peer-to-peer approach to vehicular communication for the support of traffic safety applications. In: Proceedings of IEEE 5th International Conference on Intelligent Transportation Systems, pp. 336–341 (2002)
[11] Daly, E.M., Haahr, M.: Social network analysis for routing in disconnected delay-tolerant manets. In: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing, pp. 32–40 (2007)
[12] Dikaiakos, M.D.: Intermediary infrastructures for the world wide web. Computer Networks **45**(4), 421–447 (2004)
[13] Dikaiakos, M.D., Florides, A., Nadeem, T., Iftode, L.: Location-aware services over vehicular ad-hoc networks using car-to-car communication. IEEE Journal on Selected Areas in Communications **25**(8) (2007)
[14] Dilley, J., Maggs, B.M., Parikh, J., Prokop, H., Sitaraman, R.K., Weihl, W.E.: Globally distributed content delivery. IEEE Internet Computing **6**(5), 50–58 (2002)
[15] Dow, C.R., Lin, P.J., Chen, S.C., Lin, J.H., Hwang, S.F.: A study of recent research trends and experimental guidelines in mobile ad hoc networks. In: AINA '05: Proceedings of the 19th International Conference on Advanced Information Networking and Applications, pp. 72–77 (2005)
[16] Günes, M., Siekermann, J.: Cosmos - communication scenario and mobility scenario generator for mobile ad-hoc networks. In: Proceedings of the 2nd Int. Workshop on MANETs and Interoperability Issues (MANETII'05) (2005)
[17] Hara, T.: Replica allocation methods in ad-hoc networks with data update. Mobile Networks and Applications (MONET) **8**(4), 343–354 (2003)
[18] Housel, B.C., Samaras, G., Lindquist, D.B.: Webexpress: a client/intercept based system for optimizing web browsing in a wireless environment. Mobile Networking and Applications **3**(4), 419–431 (1998)
[19] Kangasharju, J., Roberts, J.W., Ross, K.W.: Object replication strategies in content distribution networks. Computer Communications **25**(4), 376–383 (2002)
[20] Krajzewicz, D., Hertkorn, G., Rossel, C., Wagner, P.: Sumo (simulation of urban mobility); an open-source traffic simulation. In: 4th Middle East Symposium on Simulation and Modelling (MESM2002), pp. 183–187. SCS European Publishing House (2002). Http://sumo.sourceforge.net/ (last accessed January 2007)
[21] Mahdy, A.M., Deogun, J.S., Wang, J.: Mobile ad hoc networks: a hybrid approach for the selection of super peers. In: Proceedings of the 2nd IFIP International Conference on Wireless and Optical Communications Networks (WOCN 2005), pp. 280–284 (2005)
[22] Mateus, G.R., Goussevskaia, O., Loureiro, A.A.F.: Simulating demand-driven server and service location in third generation mobile networks. In: Proceedings of the 9th International Euro-Par Conference, pp. 1118–1128. Klagenfurt, Austria (2003)
[23] Miranda, H., Leggio, S., Rodrigues, L., Raatikainen, K.E.E.: An algorithm for dissemination and retrieval of information in wireless ad-hoc networks. In: Proceedings of the 13th International Euro-Par Conference, pp. 891–900. Rennes, France (2007)
[24] Morris, R., Jannotti, J., Kaashoek, F., Li, J., Couto, D.D.: Carnet: A scalable ad hoc wireless network system. In: Proceedings of the 9th ACM SIGOPS European workshop: Beyond the PC: New Challenges for the Operating System, pp. 61–65. Kolding, Denmark (2000)
[25] N. Bartolini, F.L.P., Petrioli, C.: Optimal dynamic replica placement in content delivery networks. In: 11th IEEE International Conference on Networks (ICON 2003), pp. 125–130. Sydney, Australia (2003)

[26] Nadeem, T., Dashtinezhad, S., Liao, C., Iftode, L.: Trafficview: traffic data dissemination using car-to-car communication. SIGMOBILE Mob. Comput. Commun. Rev. **8**(3), 6–19 (2004)
[27] Pallis, G., Vakali, A.: Insight and perspectives for content delivery networks. Commun. ACM **49**(1), 101–106 (2006)
[28] Pallis, G., Vakali, A., Stamos, K., Sidiropoulos, A., Katsaros, D., Manolopoulos, Y.: A latency-based object placement approach in content distribution networks. In: Third Latin American Web Congress (LA-Web 2005), pp. 140–147. Buenos Aires, Argentina (2005)
[29] Pathan, A.M.K., Buyya, R.: A taxonomy and survey of content delivery networks. Technical Report, GRIDS-TR-2007-4, Grid Computing and Distributed Systems Laboratory, The University of Melbourne (2007)
[30] Pitoura, E., Chrysanthis, P.K.: Caching and replication in mobile data management. IEEE Data Eng. Bull. **30**(3), 13–20 (2007)
[31] Presti, F.L., Petrioli, C., Vicari, C.: Dynamic replica placement in content delivery networks. In: Proceedings of the 13th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2005), pp. 357–360. Atlanta, GA, USA (2005)
[32] Rabinovich, M., Spatscheck, O.: Web Caching and Replication. Addison Wesley (2002)
[33] Rabinovich, M., Xiao, Z., Douglis, F., Kalmanek, C.R.: Moving edge-side includes to the real edge - the clients. In: Proceedings of the USENIX Symposium on Internet Technologies and Systems, pp. 12–26. Seattle, Washington, USA (2003)
[34] Ramaswamy, L., Iyengar, A., Liu, L., Douglis, F.: Automatic fragment detection in dynamic web pages and its impact on caching. IEEE Trans. Knowl. Data Eng. **17**(6), 859–874 (2005)
[35] Rilling, L., Sivasubramanian, S., Pierre, G.: High availability and scalability support for web applications. In: SAINT '07: Proceedings of the 2007 International Symposium on Applications and the Internet. IEEE Computer Society, Washington, DC, USA (2007)
[36] Roy, S., Ankcorn, J., Wee, S.: Architecture of a modular streaming media server for content delivery networks. In: Proceedings of the 2003 International Conference on Multimedia and Expo (ICME '03), Vol. 3, pp. 569–572. IEEE Computer Society, Washington, DC, USA (2003)
[37] Royer, E., Toh, C.: A review of current routing protocols for ad-hoc mobile wireless networks. Personal Communications, IEEE **6**(2), 46–55 (1999)
[38] Sailhan, F., Issarny, V.: Energy-aware web caching for mobile terminals. In: Proceedings of the 22nd International Conference on Distributed Computing Systems, pp. 820–825. IEEE Computer Society (2002)
[39] Sen, S., Rexford, J., Towsley, D.F.: Proxy prefix caching for multimedia streams. In: Proceedings of the IEEE INFOCOM, pp. 1310–1319. New York, USA (1999)
[40] Sidiropoulos, A., Pallis, G., Katsaros, D., Stamos, K., Vakali, A., Manolopoulos, Y.: Prefetching in content distribution networks via web communities identification and outsourcing. World Wide Web **11**(1), 39–70 (2008)
[41] Sivasubramanian, S., Pierre, G., van Steen, M., Alonso, G.: Analysis of caching and replication strategies for web applications. IEEE Internet Computing **11**(1), 60–66 (2007)
[42] T Nadeem, P.S., Iftode., L.: A comparative study of data dissemination models for vanets. Proceedings of the 3rd Annual International Conference on Mobile and Ubiquitous Systems (MOBIQUITOUS) (2006)
[43] Teng, W.G., Chang, C.Y., Chen, M.S.: Integrating web caching and web prefetching in client-side proxies. IEEE Trans. Parallel Distrib. Syst. **16**(5), 444–455 (2005)
[44] Tse, S.S.H.: Approximate algorithms for document placement in distributed web servers. IEEE Trans. Parallel Distrib. Syst. **16**(6), 489–496 (2005)
[45] Vakali, A., Pallis, G.: Content delivery networks: status and trends. IEEE Internet Computing **7**(6), 68–74 (2003)
[46] Wu, T., Dixit, S.: The content driven mobile internet. Wireless Personal Communications: An International Journal **26**(2–3), 135–147 (2003)
[47] Xu, Z., Guo, X., Wang, Z., Pang, Y.: The dynamic cache algorithm of proxy for streaming media. In: Proceedings of the International Conference on Intelligent Computing (ICIC 2005), pp. 1065–1074. Hefei, China (2005)
[48] Yin, L., Cao, G.: Supporting cooperative caching in ad-hoc networks. IEEE Trans. Mob.

Comput. **5**(1), 77–89 (2006)
[49] Yoshimura, T., Yonemoto, Y., Ohya, T., Etoh, M., Wee, S.: Mobile streaming media cdn enabled by dynamic smil. In: Proceedings of the 11th International World Wide Web Conference, WWW2002, pp. 651–661. Honolulu, Hawaii, USA (2002)
[50] Zemlianov, A., de Veciana, G.: Capacity of ad hoc wireless networks with infrastructure support. IEEE Journal on Selected Areas in Communications **23**(3), 657–667 (2005)
[51] Zhuo, L., Wang, C.L., Lau, F.C.M.: Load balancing in distributed web server systems with partial document replication. In: Proceedings of the 31st International Conference on Parallel Processing (ICPP), p. 305. Vancouver, Canada (2002)

第 15 章　社区网络的基础设施

Thomas Plagemann，Roberto Canonico，Jordi Domingo-Pascual，
Carmen Guerrero 和 Andreas Mauthe

15.1　引言

内容分发技术在最近的几年时间内经历了诸多变革。仅仅十年前，内容分发的主要渠道还只是通过电视和广播，而如今内容却可以通过互联网或其他的电子传输渠道以数字形式进行发布。现在，通过互联网分发多媒体内容已经得到了业界的高度重视。尽管如此，在社区范围内要想在不同种类的移动终端之间进行内容分发，仍然存在着诸多仍未解决的实际问题。早期的基于互联网的内容分发系统一般被设计成集中式系统，其中内容是通过中央服务器提供给庞大的终端用户群（见图 15.1a）。现在这种趋势正慢慢转向分布式系统（如 P2P 系统），内容提供商也已经不再仅仅局限在少数专业化的内容创作者。因此，内容分发的路径不再仅仅是直接从服务器通过主干网到达终端用户，也可以是从一个终端用户直接到达另一个终端用户（见图 15.1b）。这种趋势随着技术的发展和一些相对廉价的电子消费产品的产生而更加明显，使得人人都可以成为内容的提供者。不仅如此，高速网络的接入率（如 xDSL 网络）和 P2P 技术的普及率变得越来越高，使得计算机用户也变成了内容提供商。

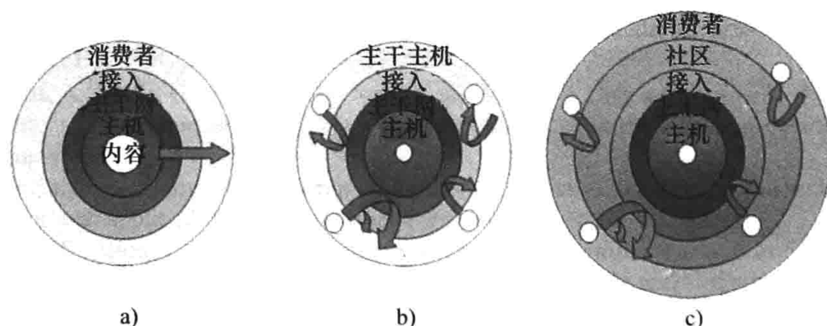


图 15.1　内容分发路径

a）传统　b）当前 P2P　c）未来连接到家庭的社区网络（基于参考文献［44］）

欧洲的“卓越网络”（European Network - of - Excellence CONTENT）[65] 在许多方面对这个领域的发展进行了深入研究，他们将重点放在内容分发在社区网络中所面临的挑战上面。在社区网络中，端用户不仅“消费内容”，而且还可以制作内容，同时为网络基础设施提供核心组件，即社区网络。这样看来，在社区网络内部，内容分发的路径也许不一定要使用由互联网服务提供商提供的任何基础设施（见图 15.1c）。

虽然“社区网络”这一术语非常直观易懂，但其具体含义仍然值得我们深入研

究，Rosson 等人[54]给出的定义如下：

“一个网络社区是指一群人通过网络通信和协作，加强和促进他们认识和目标的共同分享。网络社区的出现是底层技术发展的一个突出例子。在一个网络社区中，社区网络是物理社区与网络社区共同发展的一种特殊的情况。”

根据这一定义，社区不仅是人们通过网络协作形成的，同时也是人们通过贡献自己的资源（如个人和其邻里之间的网络）形成的。社区成员可以由几种无线网络技术提供接入网，他们也是主要通过这种方式由一个或者几个互联网服务提供商来连接网络。在社区网络中，相当一部分内容发布可以在没有任何 ISP 的参与下完成，也就是说，仅仅在物理的社区网络内部就可以完成。但是没有任何证据表明，相比传统的内容分发网（CDN）和 P2P 用户，社区网络会对互联网构成更大的威胁。

关于内容分发，最重要的概念就是在社区网络中“底层技术发展”是由“人”来推动的，这里所说的“人”也就是普通的终端用户，他们可能并没有受过特殊教育，也没有在计算机和网络管理、软件开发等方面的独特技能。因此，内容分发的离散化就必须与在所有技术层面上的自配置、自组织、自管理和自适应的解决方案相结合，只有这样才能真正地达到尽量减少内容分发过程中人工干预的目的。

此外，Cowan 等人[20]在 1998 年指明，内容服务发挥着核心作用：

“实际上，社区是那些需要被搜寻、阅读、探索、行动和更新的大量异构信息的数据仓库，并且为合作和双向交流等其他形式提供了机会。”

在 1998 年，多媒体内容还并不是这种观点的核心。然而，随着消费类电子产品和信息通信技术的发展，我们认为正是这两者使得多媒体内容的应用成为可能，并且由此产生了社区网络中对各种内容服务的强烈需求。社区成员不仅需要消费内容，而且还要同其他人分享，并要寻找特定的内容，将不同的内容进行组合，甚至自己动手编辑复杂的多媒体对象。

因此，内容的分发和使用在社区网络方面显得十分特殊。这主要有两个原因：首先，在物理社区网络中需要自治网络和覆盖网解决方案来建立和维护适当的内容分发网络；其次，人们对复杂及任意类型的内容服务（如内容的改编、转换、索引、存储等）的需求，远远超过了对内容的简单传输和消费。

为了描述社区网络的现状及其面临的短期和长期的研究挑战，本章在结构上做了如下安排：接下来的部分给出了社区网络的背景信息，包括一个简单的框架和相关的工作，随后描述了社区网络在产业领域面临的难题以及长期的研究挑战。本章的结尾部分总结了社区网络中内容分发和内容服务的若干最重要的方面。

15.2 背景和相关工作

在过去几年中一个有趣的现象是出现了大量的无线社区网络（WCN），它们向城市中的社区成员提供互联网接入。这些网络有些可能是人们自发地通过自己的各种形式的家用数字用户线路（xDSL）建立的，另一部分可能是由当地机构建立的。例如，当地议会和大学已经开始在有限区域内（如社区、校园、商业场所）或者公共建筑

内向用户群（如学生）提供无线互联网接入服务。一个“机构的”无线社区网络的例子就是阿莫斯特镇（Amherst）向它的居民提供无线网状网（Wireless Mesh Networks）[4]。

最热门的“自发”WCN由所谓的“FON社区”创建的[25]。FON成员（即Foneros）会共享他们的一些xDSL互联网连接，并由此免费接入社区的FON全球网点。当然会有某些用户并不希望与社区中的其他人共享连接，针对这种情况，FON社区也创建了一个需要付费的互联网连接业务。FON类似一种独特商业模式的Wi-Fi服务提供商。欧洲的一些商业互联网服务提供商已经开始关注涉及分享家庭互联网接入[41]的法律问题。在不久的将来，内容服务可能会被提供给社区并由此增加它的商业价值。这样一种为社区提供服务的想法已经得到一个意大利社区Ninux.org[43]的支持，该社区为其成员提供动态DNS服务和基于会话启动协议（SIP）的专用交换机（PBX）服务。

有趣的是，自发社区网络模型即使在一些不发达国家也已经被证明是成功的，尤其在为乡村地区提供互联网连接方面。自从印度解除了禁止在室外使用无线局域网的规定后，在2005年2月，Dharamsala无线mesh网社区网络就逐渐活跃起来。目前，该网络为数千用户提供宽带互联网服务。除了互联网接入，社区成员还可以通过这个网络实现文件共享、异地备份、远程高质量的影音播放和大量的网络电话（VoIP）服务。

为了应对当今和未来在社区网络中内容发布和使用的挑战，仅靠各个子系统（如内容分发网）是远远不够的。相反，包括基于IP的网络、内容分发网络、内容服务以及终端用户的整个系统都必须被包含进来。

15.2.1 架构性框架

在图15.2中描述的框架中，由于社区网络提供了基本连接，因此期望它们未来能够在框架中发挥核心的作用。在这种情况下，实际的社区网络就应该是由家庭环境、邻里之间设备构成的所有网络的集合体，并且它们的组合构成了多跳和网状网络。社区网络和社交网络类似，它的主要目的是支持本地社区。由于多媒体内容通常会分布在这样的网络中，因此就出现了一系列崭新的、吸引人的研究内容，如移动性、漫游、资源分配、用户需求/感知的体验质量（QoE）和服务质量（QoS）、拓扑鲁棒性、适应性和网络保护等。

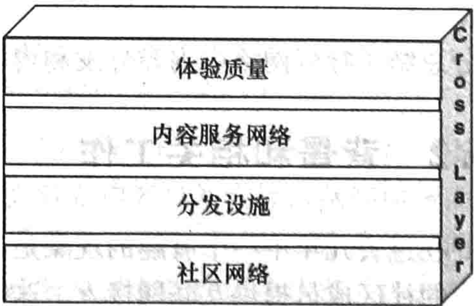


图15.2 高级架构性框架

通常，覆盖网解决方案被用于实现CDN，这是在我们架构性框架中分发设施这一层体现出来的。这些解决方案中也包括越来越多的终端用户，例如为网络提供某种资源和服务的对等点和覆盖节点。覆盖网提供了一种抽象结构，用于隐藏底层物理网络中的烦琐细节，例如构成社区网络的一个无线mesh网。然而，覆盖网也必须意识到底层

社区网络的基本属性，这样才能实现各种服务的非功能性需求，如应变能力和工作性能。覆盖网的典型功能是缓存和请求路由，它们可以通过代理缓存网络或直接连接对等节点的分布式哈希表来完成。

内容服务网络处于分发设施的顶层，它是由处理多媒体内容的一组服务构成的。这些服务应该支持音频、视频内容的整个生命周期，并且能够通过组合几个简单服务来提供更复杂的服务。内容服务的典型例子包括以下几方面：内容分类和内容抽象的自动分析和索引服务、格式变换的转码服务以及为内容搜索提供深层支持的搜索服务等。

最后，在构建框架顶部的是体验质量层，它能反映终端用户的实际体验。在一般情况下，服务质量是由一系列的技术参数定义的，这些技术参数主要捕获定量方面的内容，如吞吐量、出错率、延时、抖动、每秒传输帧数（frame/s）和每个像素的比特数。然而这些参数却并不能真实地反映用户的体验，因为用户体验并不仅仅依靠技术参数来衡量，同时还要依靠失败和错误在实际直觉上的感受。尽管体验质量是服务质量在网络、系统和应用层上不同参数的函数，但是服务质量参数和体验质量之间并不存在直接的变换关系。因此，需要确认哪一种恶化能够给用户体验带来最大的负面影响。

在社区网络中会有一些关于内容发布方面的跨层问题，这些问题横跨构建框架的这四个基本层次。这些跨层问题中有一类重要问题是关于不同层次上的服务质量参数以及参数之间如何相互关联。另一类跨层问题是关于不同层的功能可能会导致的相互影响，在自适应解决方案中，它有可能导致连锁反应或者系统行为不稳定。

15.2.2　社区网络

社区网络通常并不是网络服务提供商拥有的网络设施，相反它是由私人用户或者由分散在相对较小的地理范围内共享资源的一组用户（如邻里之间）拥有的。提供社区网络的连接是一项富有挑战性的任务，因为在节点中需要使用相当多的接入技术，并且还要求节点能有一定的移动性。

目前可能被用于社区网络设施的技术主要包括：连接到互联网服务提供商的固定节点所涉及的 xDSL、输电线路、光纤（FTTH）；无线接入网络服务供应商的节点所涉及的无线城域网（WiMAX）、移动宽带无线接入（MBWA）、3G/通用移动电话服务（UMTS）/高速下行分组服务（HSDPA）；移动节点和家庭网络所涉及的 Wi - Fi 和蓝牙技术。由于涉及如此多的网络技术，社区网络包含的节点可能不仅仅是用户终端，还可以是路由器、中继设备或网关。举例来说，固定节点可能是热点（hotspot）或同时访问点（即穿梭于社区网范围内的节点），同时也可能表现为一个移动网关、路由器或终端。

在社区网络的概念下，对多种网络技术进行了集成，如移动 IPv4 和 IPv6、多宿（multihoming,）、网络移动性（NEMO）、移动 Ad - Hoc 网络（MANET）、无线 mesh 网（WMN），以及无线传感器网络（WSN）和无线多媒体传感器网络（WMSN）等。通常，不同网络技术的相互协作并不是预先定义好或由运营者负责的，因此自治网络所具有的自我配置能力是社区网络必须具备的。总而言之，社区网络利用了各种各样

的网络技术，而这些技术造就了一个富有挑战性的研究环境。

社区网络方案中涉及的技术对接给网络带来了额外的复杂性，因为除了同种技术需要实现对接（即横向对接），还需要支持不同网络技术之间的对接（即纵向对接）。为了高效率地管理这种异构环境，美国电气电子工程师协会（IEEE）目前正在制定IEEE 802.21 标准，旨在使不同网络类型之间的对接和互用成为可能，其中就包括802 和非 802 网络。802.21 标准定义了一个抽象层，提供媒介独立对接（MIH）功能，实现对接的简化以及对不同的接入技术的管理。

由于多媒体内容的分发中包括实时分发，因此服务质量成为社区网络的一个关键问题。服务质量的配置在有线网络中仍然是一个尚未解决的问题，而在无线环境下它会更加复杂。对于这一点，在 IEEE 802.11 的扩展版本中提供可靠的服务质量管理是无线多媒体发展的关键。并且，服务质量在 MANET 和 WMN 中的作用也对社区网络中内容分发具有重要意义。

15.2.3 分发设施

在本章中所提及的分发设施是指建立在社区网络顶层的逻辑设施，其具体目的是使对内容服务的接入成为可能。目前，大多数发布设施的主要目的在于给社区成员提供高效的内容分发。为了克服传统的客户端/服务器模式的局限性，P2P 组网模式正在变得越来越普及。P2P 结构通常用于实现某种形式的覆盖网，目的是用来配置那些不能直接嵌入底层网络的服务，如组播路由对象的定位和事件传播。

请求路由和实际内容分发是通过覆盖方式实现配套服务的典型的例子。这些服务的实施可以通过与终端系统的单独协作来实现，也可以是通过专用代理服务器的支持来实现。

社区网络内容分发研究的一个共同的主题是“自治性”（autonomicity）。社区网络主要基于个人合作来给社区网络提供资源。因此，P2P 技术的重要性在于，它不仅在下载和流媒体内容的存储方面发挥着重要作用，而且在实时流媒体和资源共享等其他方面也发挥着重要作用。

在那些被广泛部署的基于 P2P 的内容分发网络中，有三个关键性组件：P2P 覆盖网络、特定的内容分发策略和缓存策略。覆盖网络负责连接参与其中的对等成员、管理对等成员的加入和离开、进行路由查询以及管理其他信息；内容分发策略负责从信息源向目的地发送所需的内容；缓存策略增加 P2P 系统中内容的可用性和有效性。

非集中式架构的巨大的潜力和优势在 Napster 上已经明显表现出来[⊖]。从那时起，大量的研究工作集中在设计自组织、可扩展、强大和高效的覆盖网络。然而需要特别指出的是，一个 P2P 覆盖网络的性能取决于多种因素，包括应用、网络节点的资源和用户行为，而这些因素在集中式系统中相关性较小。例如，一个特定的覆盖设计在

⊖ Napster 是早期一款可以在网络中下载自己想要的 MP3 文件的软件，它同时能够让用户自身成为一台服务器，为其他用户提供下载。——译者注

用户变动率较低的情况下表现得很好，但在用户变动很大时它的实施能力可能就会降低到平均水平以下。此外，内容分发系统对覆盖网络也有一定的要求，如寻找正在分享所需文件的用户、激励机制及以低成本实现高效对等通信等。因此，目前有很多研究项目和研究计划将关注点集中在内容分发系统中 P2P 覆盖网络与网络策略之间直接或间接的影响和依赖关系。

内容分发策略的很多方面都必须考虑到策略之间可能存在的相互依存关系，它们对内容分发系统的整体效率和性能的影响是至关重要的。例如，在发送文件时如何选择一个合适的调度策略就是一个重要的研究内容，BitTorrent 或网络编码使用的下载策略被证明在长期和大规模下载的情况下是一个高效的工具[26,27]。然而，随着内容分发技术的发展，也不断出现了一些新的挑战，如播客。因此，鉴于新兴的内容共享和分发策略的要求，有必要验证上述策略是否仍具有适用性。

在社区环境中，文件共享和流媒体直播应用正在迅速发展起来。这些应用和其他一些依赖连续数据流的应用（如从 IPTV 到大规模多人在线游戏）都对系统有一些特殊的要求，例如它们对延时极其敏感，需要组通信功能以及 QoS 保障。很多解决方法已经提出，但都没有得到广泛的应用。目前，为连续数据流设计的协议不再完全依赖于传统的客户端/服务器模式，可以将接收者组织成一个覆盖网络，并按照 P2P 的模式实现互相协作。

最近提出的基于 P2P 覆盖的直播音频/视频流媒体的相关方案来源于最初的将应用层组播扩展到终端系统的工作[17]。第一代控制驱动的方法重点在于根据控制平台建立一个初步的覆盖，通常是按照网状网或者树状网实施。第二代覆盖，通常是构造一个生成树来管理实际的数据传输。点对点传播（Peercast）[12]是由于流行并且拥有大量的受众而成为最有名的例子。大量的工作旨在改进点对点传播的控制平台，以解决 P2P 覆盖的高动态性，如 Nice 采用了一种复杂的聚类方案[8]。近期的研究工作都集中在使用混合树结构来改善系统鲁棒性，如 Bullet 的工作[31]。新一代数据驱动的方法强调直接处理数据的需要。对等节点之间交换可用的数据，然后根据它们需要的数据选择邻节点[8]。此外，如 epidemic 算法，Donet[63]被提出，以改善数据分发性能。可以看出，P2P 流媒体直播已经成为现实。尽管如此，迄今为止还没有证明它们在大规模数据中使用的有效性，使用仿真方法可以验证这种动态结构的可行性[50]。另一种备选方案是在实际测试平台上研究专门的应用，如 PlanetLab[46]。P2P 流媒体直播的最大规模的应用与 IPTV 相关，并涉及一些相应的专用协议和架构[41,47,48,53,56,57,58]。因此，需要进行分析的只能是它们的行为而非协议。

社区网络中对等节点的行为起了关键的作用。一端是提供资源但不求任何回报的“无私”节点，另一端是只求服务但不提供任何资源的节点，实际上这是系统中没有任何共享激励时的一种“理智”行为。因此，为了在系统和个别对等节点中达到系统资源的最佳利用，某种激励机制是必须要有的，这也是目前的一个比较活跃的研究领域。

15.2.4 内容服务网络

内容分发设施的上层是内容服务网络。一个内容服务网络是提供一系列的服务来

优化用户体验的一个网络基础设施。用户能够通过这些服务来实现轻松导航，并且根据需要获得个性化的内容。实际上，这个想法是通过将所谓的内容服务与基础网络结构相结合来提供一个内容服务的网络，并由此创建一个内容网络。内容服务网络包含大量的子区域，主要分为：

1）内容服务网络架构和服务框架，包括相关的内容服务网络的架构模型。

2）服务交互，包含服务整合和使用的所有问题，如服务查找、服务描述、服务质量、服务水平协议等。

3）服务实例，包括在内容服务网络中改善分发和用户体验的具体的内容服务。

创建一个内容服务网络的目的是希望能以开放的方式整合那些能产生和再利用资源的工具和机制，这主要是为了社区成员的利益并且允许通过创新合作进行商业应用。为了实现这一目标，就必须要有一种合适的能将内容服务轻松“插入”到服务网络中的模式和架构。最近，研究人员提出了面向服务架构（SOA）的概念，以此达到通过底层 IT 架构为业务过程提供优化支持的目的。面向服务架构的主要优势体现在部件的可重用性上，它可以很容易地组织和建立灵活、模块化的应用。因此，它似乎能够成为内容服务网络的一个合适的抽象形式。目前，面向服务的架构模式主要是使用网络服务来实现[7]。

在基于服务的内容网络架构中，主要有两个与服务交互相关的研究内容，即服务的描述方式和分发恰当服务的方式。服务描述是一个基本问题，用来确保服务能方便用户接入以及管理简单。由万维网联盟（W3C）定义的语义网㊀是这一领域比较经典的例子[2]。在不同的表达力水平上，已提出几个主要的形式，从简单的语义标记语法（如 RDF[36]）到本体（如 OWL[19]）。一个基于 OWL 的 Web 服务本体（OWL - S）已经专门为网络服务开发出来，用来清晰地描述它们的属性[37]。在最近的一个提议中定义了语义网络服务框架（SWSF）[11]，其中包括了语义网络服务语言。在服务分发领域中目前有很多工作正在进行，如 UDDI[59] 和 ebXML 注册[24]，都支持基于名称和类型的查找服务，并且根据分类进行绑定。另一个特定的成果是网络服务的动态发现（WS - Dynamic Discovery）[38]，它是基于一个本地区域的网络服务探索机制，通过本地范围内的组播来实现网络服务的搜索。

服务实例是在通信设施内部（或者边缘）提供的一些增值服务，实现对内容分发的定制和编辑。在这方面可以发掘出许多种不同类型的服务。例如，内容转换服务和质量体验服务等。内容编辑的相关问题已经研究了很长时间。例如，Smith 等人[39]提出的方法将异构终端进行针对 Ad - Hoc 的修改，他们的研究重点是对内容表示技术的定义，其中信息金字塔（Info Pyramid）起着重要的作用。Lemlouma 等人[34]为内容协商（content negotiation）提出了一种新技术。他们提出了协商和转换核（NAC），

㊀ 语义网（Semantic Web）是蒂姆·伯纳斯 - 李（Tim Berners - Lee）于 1998 年提出的一个全球信息网联盟的概念，其核心是通过给全球信息网上的文档（如 HTML）添加便于理解的元数据（Meta data），使整个互联网成为一个通用的信息交换媒介。——译者注

这是为异构终端提供协商和转换多媒体服务的一个基本系统。Lum 等人[35]强调了内容转换的必要性，并使用一个决策引擎作为逻辑实体来决定如何转换特定内容，以满足用户实际要求。Boll 等人[15]提出了一个基于强化的三阶段转换策略，即预转换、静态转换和动态转换。预转换过程中实现了一个内容的替代版本；静态转换过程中删除了不相关的替代版本；动态转换过程在前两阶段筛选出的版本中选择最合适的版本。

以前对终端用户体验的质量评价的研究工作主要适合于 MPEG－2 视频，这些方法是基于主观或者客观过程。主观方法假定人的经验作为唯一的分级因素，即质量体验；客观过程的执行没有人工干预，因此能给出更稳定的结果，但是不一定能反映用户对质量的真实看法。客观指标的例子有峰值信噪比（PSNR）、平均绝对误差（MAE）、均方误差（MSE）和均方根误差（RMSE）[52,60,61]。使用客观指标评价视频质量的方法通常不需要充分考虑人类的视觉感官，而人的感官能有效地感觉到许多明显的误差。因此，客观指标的评价结果可能不会反映用户的真实感受，这就需要使用一些结合人类视觉感官的其他方法[33,62,64]。本章的研究目标是在内容服务设施内加入用户体验质量指标，将其作为设施内的一个服务。

15.3　写给使用者

与长期研究中面临的挑战相比，业内在短期研究中的挑战是：不投机，更多地关注那些可以在未来几年内实现的技术。接下来，本节将讨论现有架构内若干方面的问题。

15.3.1　社区网络

移动性是过去几年的一个研究课题，目前已经针对 OSI 的不同层开发了相应的解决方案。具体来说，IETF 已经在 IP 层制定了移动解决方案的标准，即移动 IPv4[45]和移动 IPv6[29]。此外，它给 IPv6 制定了三个额外的标准：快速切换[30]、分层的移动 IPv6 [51]和网络移动性[22]。

大量文献[5,6]中都指出，网络中的设备在移动过程中保持连接仍然是一个挑战。此外，这些协议并没有明确支持异构网络环境。实现异构环境下的无缝切换还存在许多困难，特别是考虑到多宿主的时候。multihoming[28]技术可以提高互联网连接的可靠性，它使用多个接口连接不同的 ISP。在这种情况下，一个使用 multihoming 技术的网络节点可以使用多个不同的路径进行通信。很多文献[18,42,49]中讨论了 multihoming 技术带给静态节点或网络的益处。然而，在移动和异构网络中使用 multihoming 仍然是一个相对新颖的、面临多种挑战的研究课题。

IEEE 最近发布的 IEEE 802.21[23]旨在实现异构网络之间的交互，其中包括 802 网络和非 802 网络。IEEE 802.11e 任务组完善了 802.11 MAC，提供具有服务质量保证的音频和视频应用[9]。IEEE 802.11e 的最新批准版本给出了区分流量类别的 DCF

算法的一种改进方法。

由于 IEEE 802.11 DCF 的分布式特性，上述处于审阅中的协议也可用在多跳通信的情况下[14,16]。目前，Ad - Hoe 网和 mesh 网领域[3,10]是科学界关注的重点。通过多跳方式进行通信和路由问题是紧密相关的，且仍然是一个研究热点。在一个网络区域内，可用宽带的信息是一个重要因素，因为大部分的路由协议对 QoS 的支持都是基于该区域内的可用带宽。

最后，在这个很多问题都未解决的领域内，网络管理和设备配置主要依赖于对用户不当行为和流量异常的检测，这就使我们必须高度关注那些对网络安全性造成威胁的方面，如分布式拒绝服务（DDoS）攻击。在 MAC 层的无线 mesh 网中对异常行为的检测尤其重要，而对流量异常的检测机制则充满了整个 CDN。

15.3.2 分发设施

在改善基于 P2P 的内容分发设施时，面临的挑战在于如何创建能更好适应内容服务的特殊需求的覆盖网。

为了更好地将传统的网络内容发布至用户处，可能需要根据用户的网络位置将他们聚为一个群组。这样的一个用户群可能会有助于高效地将内容副本或代理缓存向网络中用户分布比较密集的部分移动。参考文献［32］中给出了一个实时网络用户聚类方法，从网络服务器日志中提取用户端的 IP 地址以及基于边界网关协议（BGP）路由信息的地址聚类。此类方法也许只能在传统的 CDN 上实现，因为内容提供商和 CDN 服务提供商的密切配合，使内容提供商能以最优的方式进行内容分发。

最近，P2P 模型出现在 VoIP 的应用中，如 Skype 网络电话，并在搜索用户位置和传递语音包等方面显示出很好的效果。选择一个或多个合适的对等节点来传递语音包是关系到质量、可扩展性以及 VoIP 系统成本的关键因素。然而，Ren 等人[53]的研究表明，选择对等节点需要的网络探测活动可能会影响整个应用的性能和灵活性。为了降低那些无关的 P2P 覆盖网开销，Nakao 等人[40]提出，通过在网络（底层）中提供一些基础服务，可以恰当和有效地创建那些并发的、与特定应用相关的覆盖。

P2P 应用中表现出的另一个排成是它们产生了大量流量来“逃避”ISP 对流量工程所做出的努力[55]。最近的研究[1]试图在 P2P 应用与 ISP 之间寻求一种较好的合作方式，使双方可以达到双赢的目的。

最后，目前内容分发设施的经典解决方案在设计上很好地考虑了网络的拓扑或其所处的广域网。然而，这些解决方案具有一定的局限性，因为它们并未关注客户端和终端用户群的“最后一公里”（即最终连接），而是仅仅将其视为与客户端的一个连接，并且他们并不考虑网络中内容分发设施与客户端之间连接的拓扑关系。最近的技术发展清楚地表明，相邻网络和家庭网络将会把客户端连接到核心 CDN。分发路径的改变也在相邻网络和终端用户网络中进行，并且它和广域分发设施的整合是一个尚未被系统化解决的问题。P2P 和经典 CDN 的结合似乎是一个很好的起点，Cowan 等人[20]就较早地在这一课题开展了大量的研究工作。

15.3.3 内容服务网络

适用于典型社区网络成员的内容服务主要关注的是对视听内容的消费，如 RealPlayer 和 Windows Media Player。此外，内容的提供可能是通过基于网络的服务和流媒体服务来实现的。虽然如此，用于内容创建和重新发布的服务和应用（包括管理和编辑）仍然不可用。因此，内容服务网络所提供的服务目前仍然还是局限在媒体的提供和消费上。另外，用户基本上还不能自由地添加他们自己需要的内容，也不能够创建他们自己的社区。但是，像 YouTube 和 MySpace 这样的网站已经显示出用户之间有分享信息和内容的强烈需求。然而，和这些例子刚好相反，在社区网络中有一个"目标受众"，那就是社区成员。

这暗示了，一方面有必要开发出新的服务，以提供给用户更多的自由来实现他们之间的交互和内容共享；另一方面，需要有更多的服务允许用户能够创建（新的）内容，建立他们自己的社区，并且能够由他们自己控制其环境。这超出了现有用户只能基本地管理和共享内容这一模式（如 FlickR）。在新模型中，用户还能够（在一定程度上）决定内容如何发布（如通过视频流服务），以及在资源不足的情况下该怎么办（即应该运用哪种转换策略），甚至决定应该使用哪种激励机制。因此，应该有两种不同的基本服务类型，即较为传统的内容服务以及内容和设施支持服务，特别而言，现在所提供的内容和设施支持服务还并不完善。

在这一方面，为了达到开放性和兼容性的要求，拥有一个允许不同服务提供商提供他们的内容或者支持服务的服务框架是很重要的。受网络服务的启发，需要开发一个适当的内容服务网络架构，可以用来容纳优化内容发布和使用的全部服务。这为商业和专用服务供应商提供了一个有效的机会，因为这使得它们能够在社区内容网络环境中长期提供服务。这样的内容服务网络框架为提供服务尤其是更加面向社区的服务开辟了有效的市场。为了实现这个目标，首先需要解决下一节提到的问题。

15.4 未来研究方向

本节介绍研究中面临的长期性挑战。同时，"卓越网络内容"（CONTENT Network - of - Excellence）也暴露出一些问题，本节也将介绍用于解决这些挑战的研究方向。

15.4.1 社区网络

以内容发布中的通信问题为例，其研究内容包括流媒体、网络缓存、服务质量以及 P2P 等问题。这些问题都已经得到了充分的分解，并有大量的相关研究人员从事各种子问题的研究。对于基于社区的内容网络，由于不必投资昂贵的有线设施就能够部署无线 mesh 网，因此它正变得越来越重要。然而，这个方面仍然存在大量的问题有待研究，如链路质量、信道分配、路由和网关的选择等。在将无线 mesh 网完全集成到内容网络之前，这些问题都要被研究清楚。在无线 mesh 网、网络选择和其他相关问题的研究进展基础上，可以设想到在这个年代末，集成多宿主网络将发挥其

作用。

为了实现无缝通信，需要用到端到端（End - to - End，E2E）的设施。这种设施在统一架构下集成不同的网络类型，解决如端到端服务质量、端到端服务质量路由和流量工程方面的问题。另一个研究分支则是处理异常行为检测和使内容网络远离攻击。这项研究将催生出一种对异常行为敏感的网络，使其在网络保护和检测方面具有自恢复能力。另外，并行开发是内容的自主分发，包括基于 P2P 原理的自主通信架构，其研究内容不仅涉及内容分发（如 P2P 流），也包括信任、协调和管理等方面。自主内容分发也将包括某些服务，将在下文讨论。社区内容网络内部的高互动应用仍然是一个重大的挑战，本节设想在下一个十年，基于不同网络类型的高度异构设施将能够解决这些问题，并提供必要的技术支持。

15.4.2 分发设施

未来社区网络在内容分发和服务设施方面面临的核心挑战是开发能整合自治覆盖结构和内容服务（如内容管理）的自治内容网络（CN）。自治内容网络将提高传统 CDN 的可靠性和效率，并降低管理成本。此外，自治内容网络还将扩展传统解决方案的应用领域，如向移动用户提供透明的流媒体、提供交互多媒体应用或者针对社区网络做自适应的调整。这方面的研究需要为自治内容网络设计新的架构，包括集成内容管理和内容分发的新方法以及有效传输控制信息的新协议。这里，需要解决的问题不仅涉及分发面临的实际问题，也包括如何正确地协调内容管理、服务功能和通信。后者可以通过使用跨层信息流来实现。

目前的研究重心仅仅关注于特定的应用领域，如视频点播（VoD）、IPTV 或 Web 浏览等，并且主要是面向公平地刚性分发结构。传统 CDN 中使用的方法受到相当大的限制，相对而言未来的 P2P 技术需要具有更好的自适应性，并采取更加灵活的方法。

目前的研究，尚没有系统性的解决方案，也没有一个合适的架构模型来促进相关技术发展。其中的研究的重点是如何使内容分发设施和内容管理功能能很好地同步起来。为了达到这个目标，对跨层交互的研究意义重大。这包括在通信架构不同层之间交互的功能和接口工作，目的是促进新兴的内容网络的发展和实施以及对内容创建的管理，使内容的产生更快、访问更容易。

未来的研究需要加快内容制作和分发的融合，以及弥合这两方面之间的技术差距，在此基础上，同一个框架内实现内容创作、节目形式以及端到端内容发布的可能性正在成为现实。

在改善内容分发时，一个指导原则是对其进行修改以适应网络环境，如果设计得当，就可以得到更好的系统利用率和效率。但是，如果在两个相互独立的子系统间使用这种修改版本，也就是说不采取任何进一步的预防措施，而是在自适应覆盖的顶部采用自组织的协作缓存方案，则可能会导致这样一种情况的发生：在一个层面的修改版本会阻挠在系统其他部分执行的修改版本。此外，这种状况可能促使修改版本只用到通过网络测量所产生的数据。因此，跨层的问题代表了这

方面独特的挑战。

15.4.3　内容服务网络

尽管未来需要更先进、更好的内容服务，但将现有的服务实例和更复杂的服务结合起来，共同来构建内容服务也是十分重要的。不同的服务提供商可以提供从设施支持，到实际的内容的服务。例如，前者可以，根据特定的服务水平协议（SLA）提供实时视频流设施。这项服务反过来又可以利用其他的设施服务（如 QoE 评估服务）。用户或社会团体可以租用其服务，用来实现自己内容的分发，并在它上面建立有效的内容分发服务。

为了建立这样一个框架，必须很好地定义相关的概念和基本架构来作为支撑。同时，还要为灵活的服务供应预留空间。在这样一个内部，服务本身形成了一个内容服务网络，每个服务提供一个独特的、自成一体的服务功能。可以通过一些设施来发布服务，并且形成一个协调机制的网状网，该网络既可以使用标准服务接口协调，也可以使用特定的服务协议。内容服务设计者的作用是允许不同的服务被置于整个服务框架中，使它们成为内容网络设施的一部分。在这方面的服务包括从网络架构服务（如 QoS 和 QoE 评估）到分发支持服务（如转码和内容转换），以及以内容为中心的服务（如视频摘要和索引）。

该服务的架构遵循了通用的面向服务的架构（SOA）模型。该模型提供了一种通用的服务规范，所有内容服务方面的问题都要遵守这一规范。这种规范为个性化的服务预留了足够的空间，以细化整个服务接口和功能。内容服务必须提供一系列的接口，通过这些接口与其他服务或者应用进行通信。内部组织和服务结构既不是这个模型的一部分，也不是服务特定接口的描述或服务特定功能规范的一部分。

服务可以是有状态的或无状态的。有状态服务必须为用户提供一个接口来查询其执行状态，它也应该作为一个提供服务质量协议的一部分，这需要一个服务水平协议规范。服务水平协议本身是一个具体服务，它的格式需要在服务描述中进行定义。上下文感知指的是从应用或环境上下文中获取信息来控制和管理服务。通过跨层交互，服务能够从底层和系统组件中检索信息。通过这种方式它能更好地感知系统环境，并可以适应或尝试去影响底层组件。用户交互允许由用户根据自己的偏好来制定规范以使服务能适应用户的需求。图 15.3 显示了通用内容服务模型。

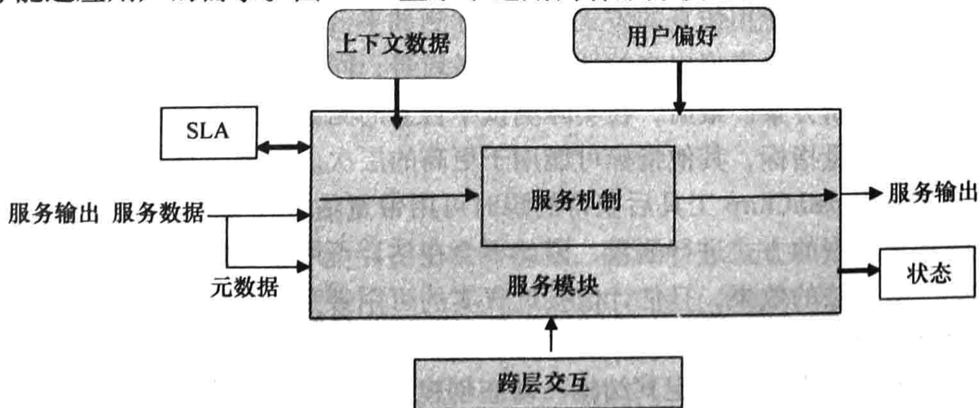


图 15.3　内容服务模型

内容服务框架提供一个语境来放置服务。服务框架内部中，至关重要的内容是服务描述及其在服务注册表中的表示。通过标准接口，服务可以经由服务注册表调用其他服务。这种方法允许内容服务之间动态、自动的组合，这为中介服务开辟了新的商业机会。

15.4.4 跨层问题

研究界普遍认为，分层的系统架构除了自身优势外也有一些明显的缺陷。为了使资源用于分布式应用，对网络层信息的访问是十分必要的。跨层的方法就是用来实现这一目标，但是它们又是为特定的解决方案而设计的。因此，了解和开发出一个比纯分层方案更好的架构是一个挑战。然而，跨层问题在未来社区网络的通信系统中特别重要，因此有必要使用自主解决方案，如自适应功能等。如前所述，不同功能的独立修改版可能会因为它们之间需要共享资源而产生相互影响，例如双方可能都会对网络流量带来影响。解决这一挑战的第一步，是为每一层确定一个衡量标准，包括服务质量参数和资源消耗指标，以及各层之间的依赖关系。虽然这一步似乎微不足道，但要想成功地实现，上述指标及其定义就必须要被这一领域的整个研究界所接受和使用。目前，正在使用许多不同的指标和定义，它们之间也互不兼容。模拟参数之间的依赖关系也需要包括对该系统要素的功能行为的理解。为了应对这一挑战，CONTENT Network - of - Excellence 开发了一个适合通信网络系统的通用基准，这是一种模块化的方法，通信网络系统的不同层可作为被测试系统，并且其他各层可以分别表示环境和工作负载。

15.5 CONTENT 方法

CONTENT 体系框架并不为基于社区的内容网络提供实施蓝图，而更多的是提供准则，且根据这种网络的发展来制定准则。为了验证所提出的准则和框架的内部机制，这个框架正在评估各个方面的内容。在这里，策略是实施的关键因素，并通过测量和仿真来评估它们的性能。该框架是在三个架构层次上实施的，或在涉及跨层活动的情况下实施的。本节将对 CONTENT 中的三个样本进行研究，通过得到的结果说明这一点。

在社区网络层，使用仿真的方式及在实际测试平台测量的方式，研究移动终端在垂直和水平切换过程中表现出来的性能和服务质量，同时仿真也用来验证无线网络中适合带宽估计的新方案。最后，在实际测试平台上的测量数据用来在网络层分析和确定适当的服务质量指标，其他指标可能用于更高的层次。作为一个初步的结果，使用本节方法及使用 pathChirp 工具后获得的瞬时可用带宽估计在图 15.4 中显示。本节研究的方法基于内嵌的方式进行测量，因此不会在估计可用带宽时引起网络拥塞。图中显示了这一新方法的效果，且估计值接近真实的可用带宽。

作为分发设施的一部分，P2P 缓存机制在本节中进行了验证。在这方面，前期工作的重点是 P2P 缓存如何确定其结构、动态创建和协调不同的元素。P2P 缓存的目标是使内容贴近用户。然而，与“常规”的缓存相比，某项内容在达到一定的访问量

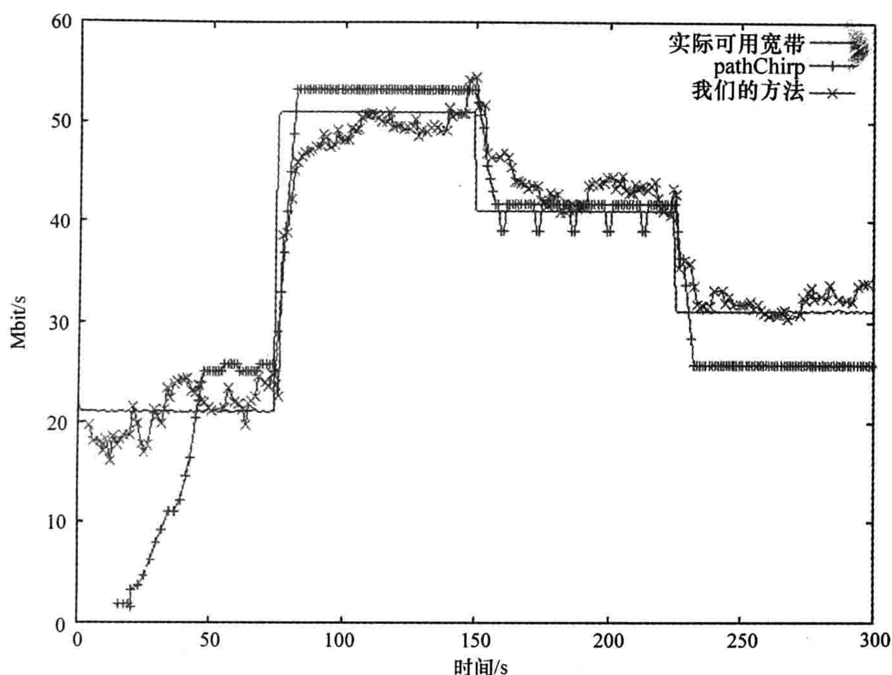


图 15.4　可用带宽估计

后就不需要缓存了，因为这时它已经广泛存在于附近的邻节点。一个想法是选择对等机（基于请求频率进行选择）来加入 P2P 缓存。内容通过基于请求的策略进行缓存，当这个内容可用之后，就从缓存中将它去除。现有的研究已在 5000 和 7000 个节点规模和不同的下载方式上进行了仿真。结果表明，主要的开销源自缓存之间的相互协调。由于内容离用户更近，因此通过节省带宽抵消了这些开销。同时也发现，可以节省的带宽量很大，而额外的开销则微不足道。但是，具体内容的大小和访问模式是至关重要的，下一步工作是研究不同下载速度和内容交叉所带来的影响。

除了使用系统仿真工具，一些原型也用于展示该研究结果在实际场景下社区网络内容分发的应用能力。根据现有的商业服务，实际场景，用来验证架构框架。我们特别研究了一个视频点播的应用场景，通过提供视频和在基于客户端的社区入口中创建 P2P 应用，社区成员和社区提供商都可以提供视频下载。这些视频可以是免费的，也可以是付费的。例如，高尔夫爱好者的社区就可以提供专业高尔夫视频之类的付费内容（如 PGA 锦标赛的录像）。它既可以提供收费的视频（如高尔夫教练录制的如何提高水平的视频），也可以提供完全免费的视频供自由下载。

15.6　结论

社区网络为新的内容服务提供了很多的机会，同时也为社区成员的沟通和互动提供了平台。在这个模型中，社区成员在网络和节点方面（例如，以无线网状网的方式）提供资源，他们提供内容、管理和使用。由于这些都是常见的终端用户，所以不需要在网络管理和系统管理方面进行特殊的训练，网络和覆盖层的自主解决方案对于减少人为干预（用于建立和维护内容分发设施）是十分重要的。在内容分发设施中，

无线网络和移动性是十分重要的部分。因此，对于可用资源进行动态、自适应的服务至关重要。为了提供自适应解决方案的理论基础，一个最重要的挑战是对跨层交互和依存关系中的功能和指标的理解和建模。此外，服务需要由若干简单的服务实例动态组合而成，这样才能根据功能需求和可用资源，准确地为用户提供所需要的服务。

致谢

本文的研究得到了 CONTENT Network - of - Excellence 的支持，并受到欧盟第六框架计划的资助（项目编号 ISTFP6 - 038423）。它是项目组所有成员共同努力的结果，因此作者尤其要感谢来自以下各个大学的同事们：University Pierre and Marie Curie（Paris 6）、University of Coimbra、National and Kapodistrian University of Athens、Technische Universität Darmstadt、AGH University of Science and Technology、Delft University of Technology。

参考文献

[1] Aggarwal V, Feldmann A, Scheideler C (2007) Can ISPS and P2P users cooperate for improved performance. SIGCOMM Computer Communications Review, Vol. 33, no. 3, pp. 29–40
[2] Akkiraju R, Farell J, Miller JA, Nagarajan N, Sheth A, Verma K (2005) Web Service Semantics - WSDL-S. W3C Workshop on Frameworks for Semantics in Web Services
[3] Akyildiz IF, Wang X, Wang W (2005) Wireless mesh networks: a survey. Computer Networks, Vol. 47, no. 4, pp. 445–487
[4] http://www.amherstma.gov/departments/Information_Technology/community_wireless.asp
[5] Aparicio AC, Serral-Gracià R, Jakab L, Domingo-PascualJ (2005) Measurement Based Analysis of the Handover in a WLAN MIPv6 Scenario. Dovrolis C (Ed.): Passive and Active Measurements 2005 Boston USA, LNCS 3431, pp. 207–218
[6] Aparicio AC, Julian-Bertomeu H, Núñez-Martínez J, Jakab L, Serral-Gracià R, Domingo-Pascual J (2005) Measurement-Based Comparison of IPv4/IPv6 Mobility Protocols on a WLAN Scenario. Networks UK, Publishers of the HET-NETs '05 Technical Proceedings (ISBN 0-9550624-0-3) Ilkley, UK
[7] Austin D, Barbi A, Ferris C, Garge S (2004) Web Services Architecture Requirements. W3C Working Group Note, http://www.w3.or/TR/wsa-reqs/
[8] Banerjee S, Lee S, Bhattacharjee B, Srinivasan A (2003) Resilient multicast using overlays. In Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pp. 102–113, New York, NY, USA, ACM Press
[9] Joe I, Batsell SG (2000) Reservation CSMA/CA for Multimedia Traffic over Mobile Ad-hoc Networks. ISCC'2000, Antibes – Juans les Pins, France
[10] Aguayo D, Bicket J, Biswas S, Judd G, Morris R (2004) Link-Level Measurements from an 802.11b Mesh Network. Proc. of ACM SIGCOMM 2004, Portland, Oregon, USA
[11] Battle S, Bernstein A, Boley H, Grosof B, Gruninger M, Hull, R, Kifer M, Martin D, McIlraith S, McGuinness D, Su J, Tabet S (2005) Semantic Web Services Framework (SWSF) Overview Version
[12] Bawa M, Deshpande H, Garcia-Molina H (2002) Streaming live media over peers. HotNets-I, Princeton, NJ, pp. 107–112
[13] Berbner R, Heckmann O, SteinmetzR (2005) An Architecture for a QoS driven Composition of Web Service based Workflows. Proceedings of Networking and Electronic Commerce Research Conference (NAEC2005) Riva del Garda, Italy
[14] Biswas S, Morris R (2005) Opportunistic Routing in Multi-Hop Wireless Networks. Proceed-

ings of SIGCOMM 2005, Philadelphia, PA, USA, pp. 69–74
[15] Boll S, Klas W, Wandel J (1999) A cross-media adaptation strategy for multimedia presentations. Proceedings of the seventh ACM International Conference on Multimedia (Part 1), pp. 37–46, New York, NY, USA
[16] Broch J, Maltz DA, Johnson DB, Hu YC, Jetcheva J (1998) A performance comparison of multi-hop wireless ad hoc network routing protocols. Mobile Computing and Networking, pp. 85–97
[17] Chu YH, Rao SG, Seshan S, Zhang H (2002) A Case for End-System Multicast. In IEEE Journal on Selected Areas in Communications, special issue on Network Support for Multicast Communications, Vol. 20, Issue 8, pp. 1456–1471
[18] Cidon I, Rom R, Shavitt Y (1999) Analysis of multi-path routing. IEEE/ACM Transactions on Networking, Vol. 7, no. 6, pp. 855–867
[19] Dean M, Connolly D, van Harmelen F, Hendler J, Horrocks I, McGuinness DL, Patel-Schneider PF, Stein LA (2002) OWL Web Ontology Language 1.0 Reference. W3C Working Draft
[20] Cowan DD, Mayfield CI, Tompa FW, Gasparini W (1998) New role for community networks. Communications of the ACM, Vol. 41, Issue 4
[21] Darlagiannis V, Mauthe A, Steinmetz R (2006) Sampling Cluster Endurance for Peer-to-Peer based Content Distribution Networks. Proceedings of Multimedia Communication and Networking (MMCN 2006, part of IS&T/SPIE Electronic Imaging 2006 Symposium)
[22] Devarapalli V, Wakikawa R, Petrescu A, Thubert P (2005) Network Mobility (NEMO) Basic Support Protocol. RFC 3963
[23] Dutta A, Das S, Famolari D, Ohba Y, Taniuchi K, Kodama T, Shulzrinne H (2005) Seamless Handover across Heterogeneous Networks - An IEEE 802.21 Centric Approach. Proceedings of WPMC2005
[24] ebXML Registry Services and Protocols, Committee Draft 01 (2005)
[25] http://www.fon.com
[26] C. Gkantsidis, T. Karagiannis, P. Rodriguez, M. Vojnovic Planet Scale Software Updates, ACM/SIGCOMM'06, Pisa. Sep 2006
[27] C. Gkantsidis and P. Rodriguez, Network Coding for Large Scale Content Distribution, IEEE/INFOCOM'05, Miami. March 2005.
[28] G. Huston, Architectural Approaches to Multi-homing for IPv6, RFC 4177, 2005
[29] Johnson D, Perkins C, Arkko J (2004) IP Mobility Support for IPv6. RFC 3775
[30] Koodli R, ed (2005) Fast Handovers for Mobile IPv6 RFC 4068
[31] Kostic D, Rodriguez A, Albrecht J, Vahdat A (2003) Bullet: high bandwidth data dissemination using overlay mesh. ACM SIGOPS Operating Systems Review, ACM SOSP
[32] Krishnamurthy B, Wang J (2000) On network-aware clustering of Web clients. Proceedings of ACM SIGCOMM
[33] Kuhmünch C, Kühne G, Schremmer C, Haenselmann T (2001) Video-scaling algorithm based on human perception for spatio-temporal stimuli. Technical Report Lehrstuhl Praktische Informatik IV, University of Mannheim, Germany
[34] Lemlouma T, Layada N (2003) Media resources adaptation for limited devices. Proceedings of the 7th ICCC/IFIP International Conference on Electronic Publishing, Universidadedo Minho, Portugal
[35] Lum WY, Lau FCM (2002) A context-aware decision engine for content adaptation. IEEE Pervasive Computing, 1(3):41–49
[36] Manola F, Miller E (2004) RDF Primer. W3C Recommendation
[37] Martin D, ed (2003) OWL-S: Semantic Markup for Web Services. Technical Overview (associated with OWL-S Release 1.1)
[38] Microsoft Corporation I (2004) Web Services Dynamic Discovery (WS-Discovery)
[39] Mohan R, Smith JR, Li CS (1999) Adapting multimedia internet content for universal access. IEEE Transactions on Multimedia, 1(1):104–114
[40] Nakao A, Peterson L, Bavier A (2003) A Routing Underlay for Overlay Networks. Proceedings of the ACM SIGCOMM Conference
[41] http://wiki.ninux.org/
[42] Ogier R, Ruenburg V, Shacham N (1993) Distributed algorithms for computing shortest pairs

of disjoint paths. IEEE Transactions on Information Theory, Vol. 93, no. 2, pp. 443–456
[43] (2006) Wi-Fi service breaches ISP conditions. OUT-LAW News, 27/09/2006. http://www.out-law.com/page-7335
[44] Parker A (2005) P2P: Opportunity of Thread. Panel presentation at IEEE Workshop on Web Content Caching and Distribution, Sophia Antipolis, France
[45] Perkins C (2002) IP Mobility Support for IPv4. RFC 3344
[46] PlanetLab: http://www.planet-lab.org/
[47] PPLive web site, http://www.pplive.com.
[48] PPStream web site, http://www.ppstream.c
[49] Raju J, Garcia-Luna-Aceves JJ (1999) A new approach to on-demand multipath routing. IEEE ICCCN
[50] Silverston T, Fourmaux O (2006) Source vs Data-Driven Approach for Live P2P Streaming. Proceedings of IEEE ICN 2006, Mauritius
[51] Soliman H, Castelluccia C, Malki EK, Bellier L (2005) Hierarchical Mobile IPv6 Mobility Management (HMIPv6). RFC 4140
[52] Stockhammer T, Hannuksela MM, Wiegand T (2003) H.264/AVC in Wireless Environments. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, Issue 7, pp. 657–673
[53] Ren S, Guo L, Zhang X (2006) ASAP: an AS-Aware Peer-relay protocol for high quality voIP. Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS'06), Lisbon, Portugal
[54] Rosson MB, Carroll JM (1998) Network communities, community networks. CHI 98 conference summary on Human factors in computing systems CHI '98
[55] Seetharaman S, Ammar M (2006) On the Interaction between Dynamic Routing in the Native and Overlay Layers. IEEE INFOCOM
[56] SOPcast web site, http://www.sopcast.com.
[57] http://tibtec.org/?q=node/60
[58] TVants web site, http://www.tvants.com.
[59] UDDI Spec Technical Committee (2003) UDDI Version 3.0.1. http://uddi.org/pubs/uddi_v3.htm
[60] Verscheure O, Frossard P, Hamdi M (1999) User-Oriented QoS Analysis in MPEG-2 Video Delivery. Journal of Real-Time Imaging, special issue on Real-Time Digital Video over Multimedia Networks, Vol. 5, no 5, pp. 305–314
[61] Wiegand T, Schwarz H, Joch A, Kossentini F, Sullivan GJ (2003) Rate-constrained coder control and comparison of video coding standards. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, no 7, pp. 688–704
[62] Winkler S (2005) Digital Video Quality - Vision Models and Metrics. Wiley
[63] Zhang X, Liu J, Li B, Yum TP (2005) Coolstreaming/donet: A data-driven overlay network for peer-to-peer live media streaming. Proceedings IEEE Infocom
[64] Wang Z, Sheikh HR, Bovik AC (2003) Objective Video Quality Assesment. In The Handbook of Video Databases: Design and Applications, eds Furht B, Marqure O, CRC Press, pp. 1041–1078
[65] CONTENT Network of Excellence, IST-FP6-038423. http://www.ist-content.eu/

第 16 章　内容分发网络的互联

Mukaddim Pathan，Rajkumar Buyya 和 James Broberg

16.1　引言

目前商业 CDN 提供商的配置方法主要是在世界各个位置放置服务器集群。然而，对于新的 CDN 提供商来说，很难提供覆盖全球的高质量服务，而且会影响现有 CDN 提供商的商业利益，从最近几年 CDN 市场主要发生的一些合并和并购现象就能明显看出这一点，其结果是市场上仅有少量巨头存活。遗憾的是，由于运营的私密性，现有的商业 CDN 提供商之间不会尝试通过合作来向终端用户分发内容。除此之外，内容提供商通常只同一家 CDN 提供商合作，这就无法同时使用多个 CDN。这种封闭和非合作的运营模式导致 CDN 提供商之间各不相同。因此，通过网络互联使不同类型的 CDN 在进行内容分发时实现协调与合作，能够促进 CDN 提供商的迅速提升，以应对瞬时拥塞[2]和需求的超预期增长，同时也可以不再依赖于某个指定的 CDN 提供商来提供资源。

除了大型企业用户，CDN 服务对其他用户的价格较为昂贵。另外，商业 CDN 通过和用户签订服务水平协议（SLA）来做出服务承诺，如果 CDN 没有履行相关服务，就会受到一定的处罚。因此，如果某个 CDN 提供的服务质量（QoS）不能满足终端用户的需求，那么它就有可能因为违反 SLA 而需要付出一定的罚金。从提供商和终端用户的成本效率和性能的角度来说，如果能够综合使用其他 CDN 未被使用的设施，就可以实现规模经济效益。为方便本章的讨论，我们将多个 CDN 之间的互联和协作技术用术语“多 CDN 对等组合”或简单的“CDN 对等”来表示，其定义为：

对等组合：“CDN 之间的对等组合是由一组自治的 CDN $\{CDN_1, CDN_2, \cdots, CDN_n\}$ 构成，它们之间根据某种机制 M 进行合作，该机制为多 CDN 合作提供实现资源共享的相关资源和设施，以确保有效的服务分发。每个 CDN_i 通过一个‘管道’C_i 和其他对等成员连接，该管道帮助该 CDN 从其他 CDN 中找到对自己有用的资源。”

虽然 CDN 对等的设想非常具有吸引力，但具体实现起来仍然需要解决很多问题。例如，需要设计一个系统，不仅可以虚拟多个 CDN 提供商，而且能够根据成本、性能和负载的情况将终端用户的请求从主提供商转移到其对等成员。特别地，需要考虑如下关键问题：

（1）什么时候对等？

也就是说，一个对等组合在什么条件下应该被激活。启动条件必须考虑到预期和非预期的负载增长。

（2）如何对等？

也就是在多个 CDN 之间形成对等组合所需要的规划（strategy）。该规划必须指定

实体之间的交互，并顾及到各对等成员采取的不同策略（policy）。

（3）和谁对等？

也就是用于选择与哪个 CDN 建立对等的决策机制。它涉及如何对对等成员性能进行预测，以及如何在面临独立管理和与对等 CDN 共享有限信息这两个问题下进行工作。

（4）如何管理和执行对策？

也就是如何根据协商的 SLA 来管理对策，包括配置必要的对策以及有效地管理这些对策。

因此，对于一个 Ad - Hoc 或设计中的对等 CDN，需要进行大量基础研究来解决如下几个核心问题：测量和传播负载信息、执行请求的分配和重定向、启用内容复制以及对分布于整个互联网尺度上的参与者给予合适的补偿。此外，为了确保 CDN 提供商之间不间断的资源共享，对等组合必须保证能为所有参与者提供足够的激励[18]。对于每一个 CDN 来说，这些问题之间存在着很强的关联性和依赖性，所以现在必须在一种协作和相互作用的方式下看待它们，同时满足施加在每个提供商上的复杂的多维约束。在接受其客户（customer）的委托代为向终端用户提供内容服务时，每个 CDN 提供商必须保证达到 SLA 的要求，同时还要履行与其他提供商约定好的相关义务。

本章提出了一种实现 CDN 对等的方法，用于构建一个“开放的”CDN，其优点是扩展性好，可以与其他 CDN 共享资源，因此它是目前已有的非协作 CDN 的升级版本。在我们的架构中，只要一个 CDN 的处理能力可以应对当前负载，那么它就向终端用户的请求提供服务。如果负载超过了其能力范围，那么处理能力之外的用户请求就会被转移到对等的 CDN。同时，我们提出了两个新的模型来支持 CDN 对等，并解决实现这些模型所需面对的问题。

本章其余内容组织如下：16.2 节描述了 CDN 对等的意义和必要性；随后介绍了相关的研究，并着重说明它们的优缺点；16.4 节提出了我们的 CDN 对等方法及一些实现辅助功能的新模型；然后，讨论了实现 CDN 对等时面临的挑战；16.7 节中讨论了需解决的核心技术问题；最后，16.8 节给出了本章的总结。

16.2 CDN 网络互联的重要性

前面几章已经提到过，由于用户请求的数量巨大，访问量大的网站通常会出现拥塞、瓶颈甚至长时间终止服务的情况。正如在第 11 章讨论的那样，当突发的瞬时拥塞出现时就会导致这种情况的发生。瞬时拥塞源于极其重大的、令人感兴趣的事件，或者当浏览像 Slashdot㊀或 Digg㊁这样的高访问量、高热门网站后导致的流量的突然攀升。

Web 服务器访问量的增长也可以具有更高的可预测性，如在奥运会和 FIFA 世界杯比赛期间。使用单一的网站甚至服务器集群都常常不能适应高访问量网站的请求规模。例如，1998 年足球世界杯的官方网站在 3 个月内收到的请求数达到了 13.5 亿，

㊀ http://www.slashdot.org。——原书注

㊁ http://www.digg.com。——原书注

其中请求数的峰值达 7300 万个/天以及 1200 万个/h[2]。1998 年冬季奥运会同样也出现了这种情况，在峰值那天官方网站处理了 5680 万个请求（最大请求量达到每分钟 110414 个）[13]。2001 年 9 月 11 日，许多主流新闻网站服务器的可用性几乎为 0，完全加载一个页面需要 45s 以上[15]。即使终端用户在终止访问前愿意等待的时间小到仅仅 10s，也将会导致额外的宽带和资源浪费[12]。

对等 CDN 也许可以成为一个好的办法来解决瞬时拥塞、Web 资源过度配置和不利情况的影响。很明显，如果存在一个框架，它能够在 CDN 之间实现对等，并使服务器负载以协作的方式进行共享，那么成本效益、性能、可扩展性和服务覆盖范围都将得到显著的提高。为了更好地理解 CDN 对等，先以图 16.1 中显示的情况为例。

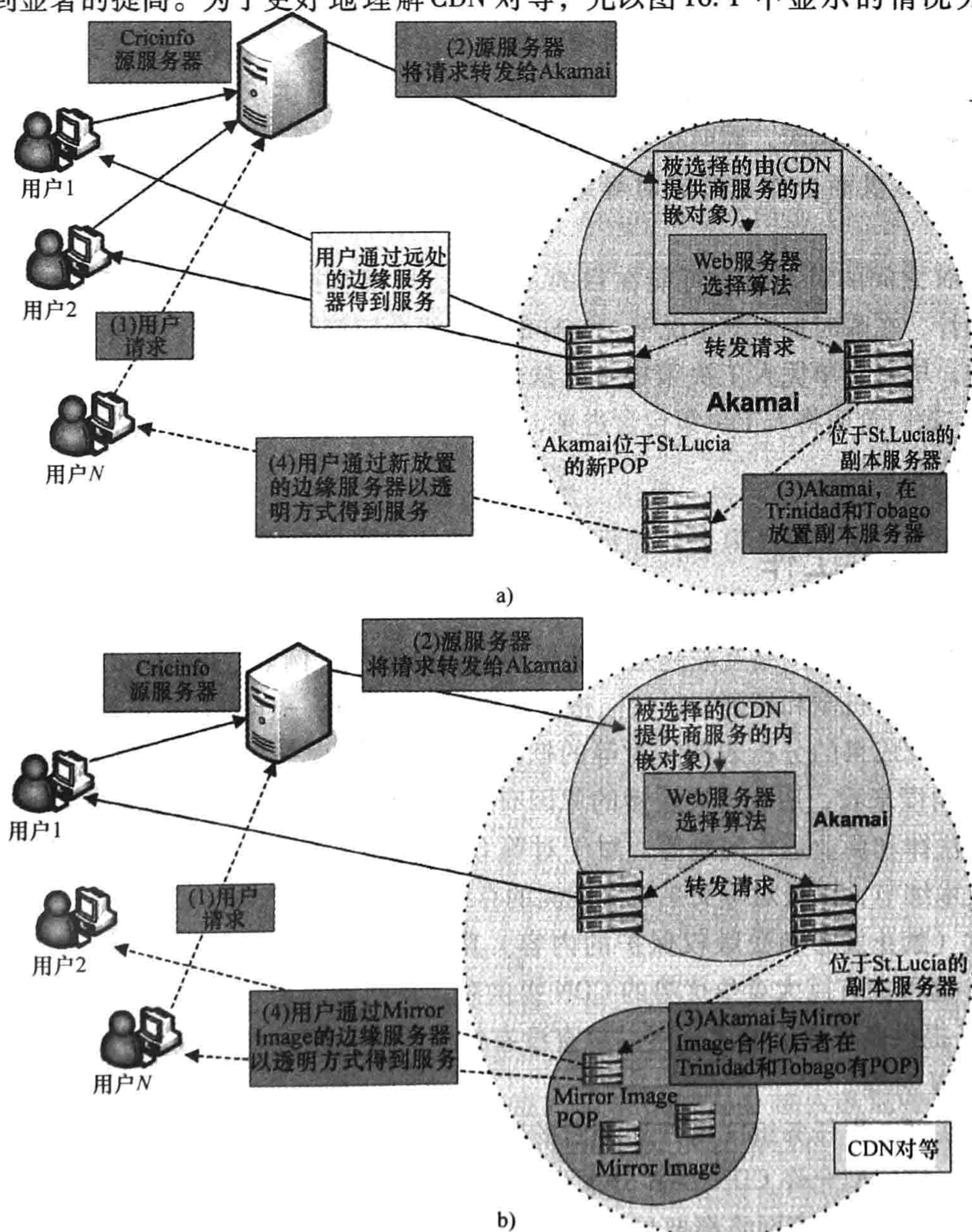


图 16.1　一个 CDN 对等

a）CDN 提供商通过放置一个新的 POP 进行扩展　b）CDN 提供商之间的对等

假设要在加勒比海举办 ICC 板球世界杯赛，提供直播报道的网站为 www. cricinfo. com。这个网站作为一个内容提供商，与 CDN 提供商 Akamai[10] 之间专门签订了一个 SLA 协议。大部分的比赛要在特立尼达和多巴哥岛（一个加勒比海岛）举行，但 Akamai 在这里并没有接入点（POP）。

作为板球比赛的东道主，当地人无疑会对 www. cricinfo. com 提供的现场直播抱有极大的兴趣。假设 Akamai 事先已预料到了这种状况，那么 Akamai 的管理人员可以采取必要的措施来应对各种突发情况。为了给用户提供更好的服务，Akamai 有两种选择：可以在特立尼达和多巴哥群岛设立代理缓存服务器（surrogate），或者也可以使用距离这里较远的其他边缘服务器，如图 16. 1a 所示。然而必须指出，首先，仅仅为了一个特定的事件而设立新的代理缓存服务器将带来极高的成本，并且事件结束后这个服务器可能就没用了；其次，如果不能为用户请求提供承诺的 QoS，那么 Akamai 的声誉就会面临风险，同时也会违背 SLA 并造成利润的降低。因此，Akamai 可以与其他 CDN 提供商合作，形成对等组合来分发那些它无法分发的服务，如图 16. 1b 所示。

CDN 之间的对等组合可能在目的、范围、规模和持续时间等方面都存在差异。我们预计，在瞬时拥塞的情况下，这样的对等组合应该能够在一个时间上限内自动做出反应。毕竟，单凭人工决策根本无法快速应对多变的情况。对于长期事件（见图 16. 1），我们希望将人工决策包含进来，这样才能保证由此产生的任何决策能够顺应企业的战略目标。

16. 3 相关工作

资源提供商的网络互联越来越受到研究者的重视。例如，InterGrid[3] 描述了网络互联的架构、机制和原则，以便网格能够以类似互联网一样的方式进行扩展。然而，对先前研究成果的分析表明，对等的框架和策略等方面并没有取得很大的进展，从 CDN 的角度来看，造成进展缓慢的原因有很多，包括需要解决的技术问题以及 CDN 自身的法律和商业运作问题等。对于对等 CDN 而言，需要一个共同的协议来定义交互的技术细节，以及对等关系建立期间的存续期和 QoS。此外，通常还存在烦琐的法律条款（禁止发布和受版权保护的内容）阻碍 CDN 之间的自由合作。最后，对于 Akamai 这种拥有巨大竞争优势的 CDN 提供商，其网络的地域覆盖范围比任何其他商业 CDN 提供商都要大的多，所以简直没有任何令人信服的商业理由来让这种提供商加入 CDN 对等。

互联网工程任务组（IETF）制定的新的互联网草案提出了内容分发互联（CDI）模型[9]，该模型允许 CDN 将其分发设施连接到其他有内容要分发的 CDN 上。根据 CDI 模型，每个内容网络都把与其相邻的内容网络看做黑盒，黑盒之间使用常规协议进行内容的网络互联，而在黑盒内部则使用自己专有的协议。因此，互联的 CDN 仍能隐藏各自的细节部分。CDI 互联网草案提出了 CDN 之间的互联，但是并没有清楚

地解释这种互联如何建立，以及互联之间的关系特性。

CDI 的一种协议架构[21]试图实现在独立管理的 CDN 之间的交互和合作。在这个框架中，一个处于管理角色的 CDN（相对于某一个特定的组而言）在转发请求之前，会在 CDN 之间交换性能数据，这在客户端响应时间上增加了一个额外的开销。此外，作为端对端协议，如果一个端点停止服务，那么在它恢复工作之前连接将一直是中断的。由于不存在对性能数据交换的评价机制，所以该协议的有效性目前还不清楚。

CDN 中介（CDN brokering）[3]允许一个 CDN 将终端用户动态、智能地重定向到域中的其他 CDN。这种基于 DNS 的系统叫做智能域名服务器（IDNS）。该系统存在的缺点是其机制在本质上是专有的，因此它可能不适用于通用的 CDI 架构。尽管它能带来更高的 CDN 容量、更低的成本以及更好的容错能力，但在处理请求时并没有明显地顾及到终端用户的 QoS 体验。此外，IDNS 看起来更像是证明了中介概念的可行性，而不是对一个特定 CDN 的性能进行综合评价。

针对多提供商架构中的内容分发载荷问题，Amini 等人[1]给出了一个对等系统。该系统的核心部分是一个对等算法，该算法将用户请求重定向到其他提供商，以实现成本的最小化和提高性能。然而，Amini 等人并没有对提供商之间的对等策略、资源配置和 QoS 担保等方面进行研究。

从用户的角度来看，协作联网（CoopNet）[15]可以使终端用户之间实现合作，由此带来了网络性能的改善。一旦发生瞬时拥塞，就可以启动这种用户之间的协作。我们发现，CoopNet 对资源有限的小型网站比较有效。但是，这种基于用户端的机制存在的最大问题是它们对于用户来说是不透明的，这可能会使其无法得到广泛使用，因而无法取代基于架构的 CDN 之间的合作方式。

CoDeeN[16,23]提供的内容分发服务完全由终端用户的需求驱动。然而，使用 CoDeeN 服务时对终端用户也是不透明的，因为这时需要用户亲自使用手动方式在浏览器的代理选项中进行设置后才能够与 CoDeeN 实现交互。这种由用户驱动的方式意味着 CoDeeN 在本质上只是一种精巧的缓存机制，而不是一个真正的 CDN。作者也曾注意到该系统很可能会被带宽占用（bandwidth hog）、密码破解和授权内容盗窃等恶意行为所滥用，这就需要 CoDeeN 具备一些基本的保护措施，如 IP 黑名单、本地与外部用户的权限分离等。目前，CoDeeN 只运行于 PlanetLab 上的节点，其作者也在设想与外部的内容提供商合作，但目前仍尚未实现。

CoralCDN[11]提出了一种新的 P2P DNS 方法，可以将用户转到 CoralCDN 覆盖网中的副本节点，从而降低源服务器的压力并提高用户的性能体验。CoralCDN 是一个协作网络，但是没有提供任何手段来使新的节点（或者提供商）加入其对等网，也不支持与 PlanetLab 外的节点实现网络互联。而目前网内的已有节点也仅仅使用了一个比较粗略的控制方法（如允许单个服务器指定其最大峰值和稳态带宽使用量），不仅没有给节点提供精细粒度的控制策略（例如一个节点为哪些内容提供服务），也没有任何对服务质量的担保。当然，考虑到该服务是免费且面向研究的，这种状况就可以理解了。该服务基于“尽力而为”的机制提供内容服务，且没有给参与协作的节点

任何补偿。

Globule[19,20]是一个开源的协作式CDN，它允许几乎任何一台托管Web内容的服务器在安装定制化Globule Apache软件后加入对等，之所以选择Apache是因为它是一个被广泛应用的主流Web服务器平台。Globule能够实现服务器之间的对等、Ad-Hoc选择、副本的创建和销毁、副本的一致性管理以及将客户端相对透明地重定向（通过HTTP或者DNS）到高性能副本。Globule CDN中参与对等的服务器既可以是托管服务器也可以是被托管服务器，或两者均可。这意味着除了能为它们自己的站点和其他用户站点提供内容服务外，它们还能够通过平衡其他参与者的资源来复制自己的站点。带宽和资源的约束可以应用到被托管服务器，但是这并不是由Globule自己来处理的，而是要依靠托管服务器中可用的相应设施来强化这个约束（如利用可限定带宽的Apache模块和“监管”环境来限制资源使用）。参与者可以使用Globule提供的一个代理服务来注册和访问其他参与者的资源，以便启动针对托管请求的协商。这些协商包括定价和补偿协议，但在Globule中这些问题并没有得到深入研究。Globule CDN也考虑到了安全性和数据完整性问题（如对恶意用户的相关处理），但实现的还远远不够。

DotSlash[25]是一个基于社区的“互助”服务，可以帮助一些没有足够资源的小型站点应对瞬时拥塞问题。只要存在这种问题的站点通过一定的配置接入DotSlash，那么当瞬时拥塞发生时该服务就能自动地干预：根据负载的情况分配和释放服务器，并在瞬时拥塞消失后停止救援。参与者可以使用服务目录来方便地实现相互定位。DotSlash中的参与者有三种固定和互斥的状态：SOS状态、救援状态和正常状态。SOS状态表示该参与者目前过载，并从其他参与者那里接受救援；救援状态表示该参与者正在向那些处于SOS状态的参与者提供救援。由于DotSlash的本质是基于社区驱动，所以没有向互联节点提供任何设施来接受参与对等组合的补偿（无论资金类还是资源类）。

16.4 CDN网络互联/对等的架构

不同CDN之间的网络互联仍然是一个远未解决的问题。通过对等机制进行CDN网络互联的概念具有很大的吸引力，人们希望通过对其研究，可以解决突发的瞬时拥塞问题，以及对短期或长期的网络请求增长进行预测，尤其是当单个CDN的资源较为匮乏时。通过形成对等组合，那些在某个特定地区不具备资源的CDN就可以利用其他CDN的资源，解决内容需求增长的局部性问题。然而，正如16.3节所述，目前协作式CDN虽然很多，但其运行却相互独立，并且不同的商业CDN都是依据不同的决策、方法和QoS水平来运营。因此，为了使这些相互独立的CDN实现对等，我们需要对对等进行规范化，包括对等方式、如何相互作用以及如何设置和管理QoS水平。

在之前的工作中[5,17]，我们已经提出了一个由策略驱动的对等CDN框架（见图16.2）。表16.1中列出了用来描述这个系统架构的术语。对等协商的发起者称为主CDN，其他同意提供资源的CDN称为对等CDN。两个CDN之间对等协商的目标称为

合约（SLA），它标识出那些被分配的用于为主 CDN 提供内容服务的对等资源（Web 服务器、带宽等）。主 CDN 管理它所得到的资源，并决定多大比例的 Web 流量（即终端用户请求）被重定向到对等 CDN 的 Web 服务器。

图 16.3 列出了创建一个对等组合的主要步骤。我们将这些步骤总结如下：

1）对等组合的创建开始于主 CDN 提供商意识到它无法处理其 Web 服务器中的一部分工作负载，此时主 CDN 向调解器（mediator）发送一个启动请求。

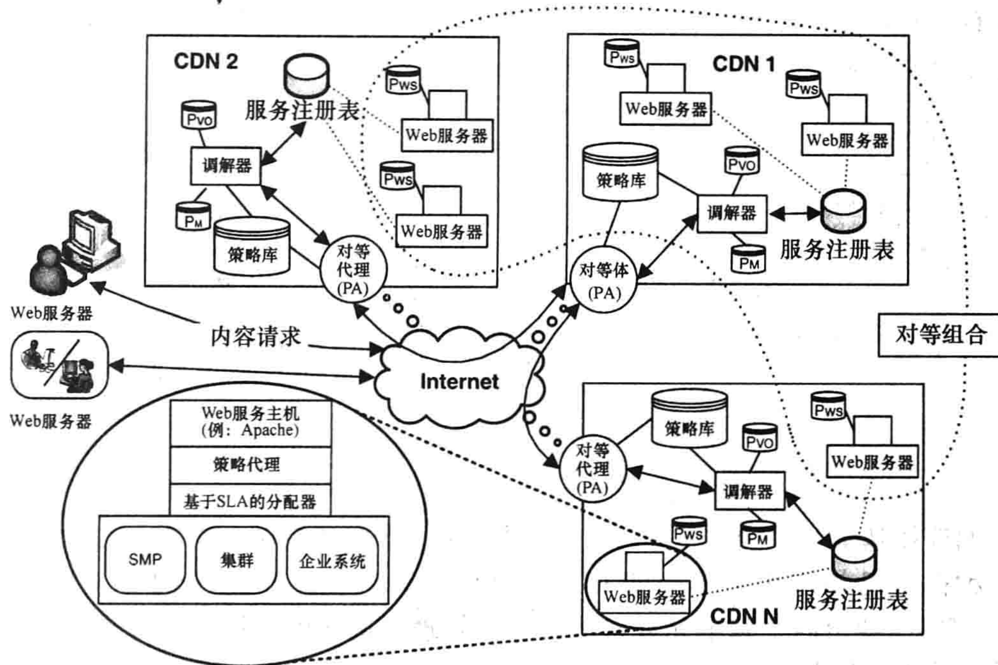


图 16.2 帮助创建对等 CDN 的系统架构

表 16.1 常用术语

术　语	描　述
Web 服务器（WS）	一个内容的存储器
调解器	一个由策略驱动的实体，被授权进行策略的协商和管理
服务注册表（SR）	在本地域内搜索和存储资源和策略信息
对等代理（PA）	对等 CDN 环境中的一个资源搜索模块
策略库（PR）	Web 服务器、调解器和对等策略的存储空间
P_{WS}	用于内容存储和管理的，与特定 Web 服务器相关的规则集
P_{M}	用于交互和协商的，与特定调解器相关的规则集
$P_{Peering}$	用于创建和发展对等组合的规则集

2）调解器实例从服务注册表获得资源和访问信息，从策略库得到 SLA 及其他策略。

3）代表主 CDN 的调解器实例根据当前状况和客户的 SLA 需求，产生服务需求，因此它需要进行扩充，以包含来自其他 CDN 的额外资源。

4）调解器实例将服务需求传递给本地的对等代理（PA）。如果当前有对等组合

存在（对于长期方案），那么立刻将其返回给主 CDN；否则，与 PA 指定的对等 CDN 进行短期协商。

5）当主 CDN 从对等处获得足够的能满足客户 SLA 需求的资源时，新的对等组合就开始运作。如果没有 CDN 对建立对等感兴趣，那么算法转回到第 3 步，重新确定服务需求，通过重新协商来创建对等组合。

如果以下情况发生，那么现存的对等组合可能需要解散或者重新建立：①形成对等的条件不再存在；②对等不再对参与的各 CDN 有益；③一个现存的对等组合需要进一步扩展来处理额外的负载；④参与的各 CDN 没有履行自己承诺的贡献。

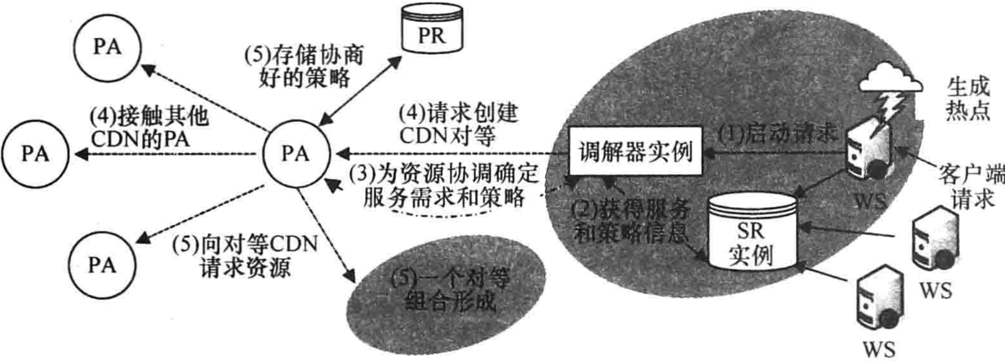


图 16.3 创建一个对等组合的典型步骤

我们选择基于 IETF 策略的框架，以便实施、管理和控制对网络资源的访问[24]。尽管这种框架已初步用于单个的 CDN，但目前尚未在多个对等的 CDN 中得到应用[22]。策略框架由四个基本元素构成：策略管理、策略库、策略实施点（PEP）以及策略决策点（PDP）。

在标准的 IETF 策略框架中，管理域是指一个在系统边界内实施、管理和控制访问资源的实体。管理员使用策略管理工具来定义系统中要实施的策略。PEP 是系统边界内的逻辑实体，负责执行被定义的策略，执行结果对系统本身有直接影响。PEP 需要遵循的策略存储在策略库中，管理员使用策略管理工具产生这些策略。PDP 负责从策略库中检索策略、解释策略（基于策略条件）以及决定 PEP 要执行哪个策略（即策略规则）。如何将这些逻辑元素配置在 CDN 系统中的恰当位置，显然对相关 CDN 和终端用户所体验到的功能和性能会产生巨大的影响，因此需要根据其所在的特定 CDN 平台，认真地加以考虑。

在对等 CDN 中，策略可以是对等 CDN 中的参与者共同认可的那些共识。这些共识定义了什么类型的内容和服务可以发布到 CDN 节点、参与者之间可以共享什么资源、采取什么指标来确保基于 SLA 的 QoS，以及在源服务器处必须执行什么类型的程序和数据。

本节提出的对等 CDN 模型如图 16.2 所示，它被映射到 IETF 策略框架，见表 16.2。策略库负责存储由对等组管理员使用策略管理工具产生的策略，一个典型的例子是用于建组的“发起者”策略。策略库对 Web 服务器、调解器和对等策略进行虚拟化，这些策略也是由策略管理工具产生的。分发网和 Web 服务器组件（即 Web 服务主机、策略代理、基于 SLA 的分配器）是 PEP 的实例，负责执行存储在策略库中

的对等 CDN 策略。对等代理和调解器是 PDP 的实例，它们指定了在与其他 CDN 协作时需要协商的一系列策略，这些策略在协商时将传给对等代理。策略管理工具由管理员负责，随 CDN 平台的不同而有所区别。使用这种基于策略的架构，其直接的好处是能够通过一个共同的对等框架促进 CDN 之间的协作能力，最终降低 CDN 的运营成本，使 CDN 在超负载条件下仍可以满足终端用户的 QoS 需求。

表 16.2　策略映射

策略框架组件	对等 CDN 组件	特定策略	描　述
系统	对等 CDN	系统中所有策略	分布式计算和网络架构（用于对等 CDN）
管理域	对等组合	经过协调的对等策略	一个用于资源管理和访问控制的管理实体
策略管理工具	与管理员有关	—	一个管理员相关的工具，用于产生策略
策略库	策略库	Web 服务器对等和调解、策略	系统中的存储策略
策略增强点	Web 服务主机、策略代理、基于 SLA 的分配器	Web 服务器策略	一个逻辑实体，确保对策略的恰当增强
PDP	调解器	调解器策略 对等策略	一个授权实体，用于检索库中的策略

我们提出了基于排队论原理的性能模型，以验证 CDN 之间对等的效果，并描述一个 CDN 的 QoS 性能特性。

设终端用户的 N 个独立的请求流到达一个称为分派器（dispatcher）的概念实体，它们服从平均到达率为 λ_i（$i \in \{1, 2, \cdots, N\}$）的泊松过程。分派器在一个特定的对等关系中起到一个集中调度器的作用，它采用独立机制，以对用户透明的方式在相关 CDN 之间分发内容请求。如果被接收到的用户请求不能被 CDN_i 服务，那么该 CDN 将这个请求重定向到其对等的 CDN 处。由于这种分派是作用于 Web 内容的个别请求上，因此它的控制级别能够达到精细的粒度。分派器遵循某个特定的策略来将部分请求从 CDN_i 分配到 CDN_j。

在实验部分，我们考虑了一个已经建立的由三个 CDN 组成的对等组合。假设 CDN Web 服务器的总处理能力是累加的，并且每个对等都包含了相同的内容副本。每个 CDN 处理能力的服务时间服从一般分布。此处利用术语“任务”来表示广义的到达服务请求。我们将一个到达的请求处理需求情况定义为“任务大小”。每个 CDN 被建模为一个随着任务大小变化而高度变化的超指数分布的 M/G/1 队列，该分布近似于一个长尾有界的 Pareto 服务分布（α，k，p）。因此，工作负载模型结合了网络访问的高变化性和自相似性的特点。

在我们的性能模型中，参与的内容提供商根据非抢占排头（HOL）优先级排队系统进行组合提供。HOL 是一个 M/G/1 排队系统，假设用户的优先级在它们到达 CDN 时已知，这样它们能够立刻以有序的方式进入队列。因此，根据已知的优先级关系可以区分不同服务和请求的各种优先级。在我们的模型中，一个输入请求（优先级为 p）加入队列内，其前面是所有优先级小于等于 p 的用户请求，其后是那些优

先级大于 p 的用户请求。由于 CDN 模型的这种性质，对等的效果不会受到特定的请求重定向策略的影响。

实验中，我们认为平均等待时间是评估 CDN 性能的一个重要参数。平均等待时间对应于用户请求被 CDN 服务之前所经过的时间。在我们的对等方案中也假设了一个 SLA，即主 CDN 需要在不超过 20000 个时间单位内服务所有用户的请求。

1. 主 CDN 的 QoS 性能

首先，我们提供的证据表明 CDN 之间的对等组合能够协助主 CDN 向用户提供更好的 QoS。主 CDN 的等待时间的累积分布函数（CDF）可以用来作为 QoS 性能指标。在对等 CDN 这样一个处于高度变化的系统中，CDF 的值明显高于均值。

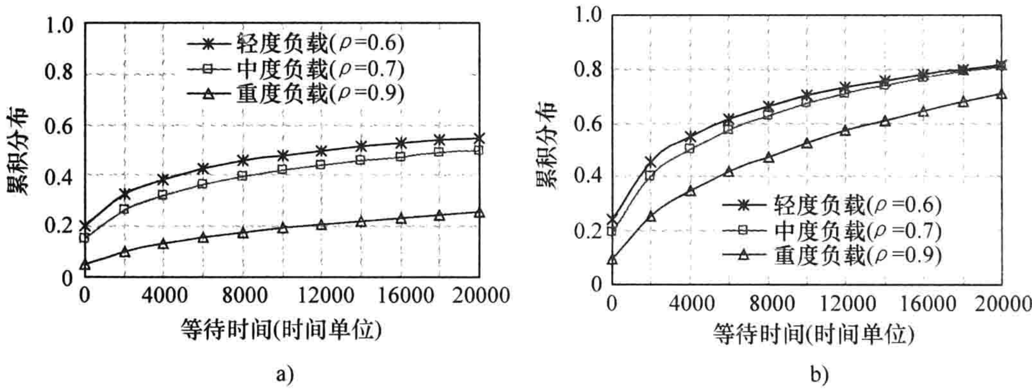


图 16.4 主 CDN 等待时间的累积分布

a）没有对等 b）处于对等组合中

图 16.4a 显示了主 CDN 在未对等情况下不同负载对应的等待时间的 CDF。从图中可以看出，设时间阈值为 20 000 个时间单位，对于轻度负载 $\rho=0.6$，用户等待时间低于阈值的概率大约是 55%；对于中度负载 $\rho=0.7$，该概率是 50%；而对于重度负载 $\rho=0.9$，该概率高于 24%。

图 16.4b 显示了主 CDN 在对等情况下不同负载对应的等待时间的 CDF。通过比较图 16.4a 和图 16.4b 可以看出，对于轻度负载 $\rho=0.6$，用户等待时间低于阈值（大小同上）的概率大约是 80%。因此，在我们的方案中，通过对等的帮助，主 CDN 的 QoS 性能提高了大约 31%。对于中度负载 $\rho=0.7$，等待时间低于阈值的概率大于 81%，提高了 38%。对于重度负载 $\rho=0.9$，概率大约是 70%，提高了 65%。而且，对于 $\rho>0.9$ 的负载，相信性能模型会有更大的提高。基于这些观察，我们可以认为，在不考虑任何特定的重定向请求策略的情况下，相比于非对等情况，CDN 之间的对等策略实现了 QoS 性能的大幅度提高。

2. 请求重定向的影响

现在来研究请求重定向策略对主 CDN 用户平均等待时间的影响。重定向策略用来决定哪些请求需要重定向到对等成员处。在前面章节中已经为对等 CDN 模型评估了不同的请求重定向策略。这里仅需要给出在所有的对等 CDN 中使用均匀负载均衡（ULB）的请求重定向策略时的性能结果，也就是说，在所有的对等 CDN 中间以相同的概率对重定向的内容请求进行分配。我们的目的是表明，与非对等结果相比，即便

使用简单的请求重定向策略，性能模型也能对平均等待时间产生显著的改善。

实验中，假设直到主 CDN 的负载达到阈值时（$\rho=0.5$）才开始使用重定向，这个负载值也用做负载基准来比较在不同主 CDN 负载时的等待时间。超出这一负载的部分将会分流给对等成员，每个对等成员只准备好接收重定向请求中的一部分（按接收阈值计算）。到达特定对等成员的重定向请求如果超过接收阈值，则被丢弃以维持系统的平衡。我们考虑轻负载的对等（对等 1 和对等 2 的负载分别设置为 0.5 和 0.4），同时调节主 CDN 的负载（$0.1\leqslant\rho\leqslant0.9$）。需要注意，为了得到请求重定向的效果，我们给出的是等待时间的加权平均值。

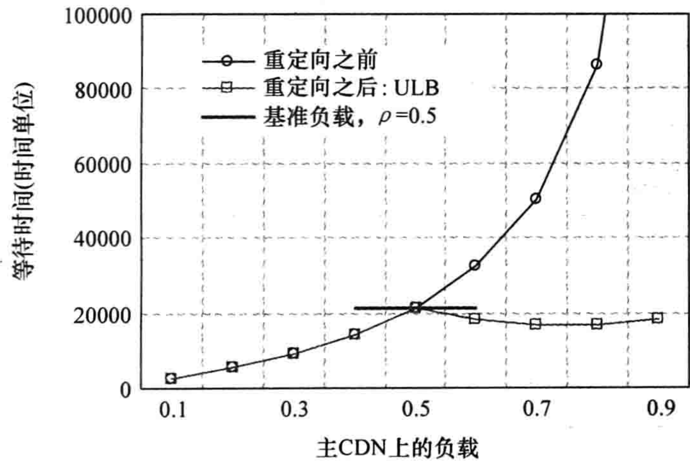


图 16.5　均匀请求重定向策略下，请求重定向对主 CDN 等待时间的影响

从图 16.5 中发现，不使用请求重定向时，当主 CDN 的负载达到 1.0 时，用户感知的主 CDN 服务性能（即等待时间）趋于无穷大。另外，使用请求重定向之后，主 CDN 的等待时间降低了，这是由于请求被重定向到对等成员处。从图中可以看出，对于轻度负载 $\rho=0.6$，等待时间减少了 43%；对于中度负载 $\rho=0.7$，等待时间减少了 66%；对于重度负载 $\rho=0.9$，等待时间减少了 90% 以上。从结果可以看出，即使像 ULB 这样简单的请求重定向策略也能保证最大等待时间低于 20000 个时间单位（按照 SLA）。因此，可以预测，一个可扩展的高效请求重定向策略可能会带来更好的性能。我们的结果也表明，仅重定向一小部分请求就能够减轻对等系统中的不稳定性和过载问题，各对等成员也避免了被突发的请求拖垮。

16.5　CDN 对等的新模型

在本节中，我们提出两种辅助 CDN 对等的新模型，它们分别是基于中介的模型和由 QoS 驱动（定制的）的基于中介的模型，可用于完善在 16.4 节中提出的对等 CDN 模型。为了更好地理解这些模型的独特之处，并将它们与已有的模型做对比，首先回顾基于 P2P 的传统网络互联/对等 CDN 模型，然后提出我们的用于形成 CDN 对等的新设想。表 16.3 对已有 CDN 模型和本章提出的 CDN 模型进行了对比，并总结了它们各自的特点。

表 16.3 CDN 模型的对比

特性	典型的 CDN 模型		用于 CDN 对等的高级模型		
	传统 CDN	基于 P2P 的 CDN	对等 CDN	基于中介的模型	由 QoS 驱动(定制)的基于中介的模型
内容分发的本质	基于 Web 服务器的协作	基于对等和内容的可用性	基于 CDN 网络互连/对等	基于 CDN 的性能	基于用户定义的 QoS(定制)
内容有效分发的责任方	CDN 提供商	对等/用户	主 CDN	内容提供商	内容提供商
协议中的实体	CDN 与内容提供商	没有实际的协议（自利的用户）	CDN 与内容提供商，CDN 与 CDN	CDN 与内容提供商	CDN 与内容提供商
协议的本质	静态	N/A	短期或长期	基于策略	动态
扩展性	有限	高	高	高	高
与外部 CDN 的协作	否	否	是	是	是
CDN 之间的协作	否	否	是	否 各 CDN 并行工作	否 各 CDN 并行工作
用户之间的协作	否	是	否	否	否

16.5.1 已有的 CDN 模型

在传统的 CDN 模型中，终端用户从特定内容提供商的 Web 站点上请求内容。而实际内容本身是由内容提供商租用的 CDN 提供，这些 CDN 从距离终端用户最近的边缘服务器上将内容发送给用户。通常内容提供商和 CDN 提供商之间会就内容提供商所能提供给终端用户的服务标准达成协议，包括保证运行时间、平均延时以及其他参数。传统 CDN 的典型例子包括 Akamai、Limelight Networks 和 Mirror Image 等。它们是典型的单一实体，不会彼此合作来发布内容和履行服务义务。这种模式更适合于那些在全球各处都有服务设施的 CDN 提供商，能够面向大多数用户来配置边缘服务器，并拥有足够的能力处理（由瞬时拥塞造成的）峰值负载。尽管 CDN 之间没有合作，但是一个 CDN 内部的 Web 服务器之间会进行合作（协作式内容分发），以保证内容按需复制并满足所有的 SLA。因此，高效内容分发的责任完全落在单个 CDN 提供商的身上。

在基于 P2P 的 CDN 中，内容提供商利用（全部使用或者仅仅作为传统 CDN 的辅助部分）端用户节点来及时有效地分发内容，如 CoDeeN、Coral 和 Globule。前两者作为志愿者节点被部署在 PlanetLab 平台上，而 Globule 则运行在终端节点上。CoopNet 和 DotSlash 是另外的两个例子，CoopNet 允许终端用户在瞬时拥塞出现时通过合作来提高用户的性能体验，DotSlash 是一个由社区驱动的互助服务，用来缓解瞬时拥塞。在这种 CDN 中，终端用户能够相互协作来提高所有用户的性能体验，尤其是在同一

地理区域的同一边缘服务器附近的众多用户能通过之间的协助从其他用户处接收内容，这种协作要能在需要的时候（如突发访问时）被动态触发。目前，在对参与协作的终端用户的贡献度的定义上还不存在任何共识，所以内容提供商难以执行特定的 QoS 指标。假设用户都是参与这种对等组合但无相应补偿的自利实体，那么它们仅仅在内容适合自身时才执行内容分发。

在网络互联/对等的 CDN 中，内容提供商租用特定 CDN 提供商的服务来向终端用户分发内容，这一点与传统的 CDN 相同。而受内容提供商委托的 CDN 提供商可以与其他的 CDN 建立对等，以协助它进行内容的分发，并满足它与内容提供商签订的 SLA。对等 CDN 的例子包括 IETF CDI 模型[9]、CDN 中介[3]、多提供商内容分发服务的对等[1]以及我们提出的对等 CDN[5,17]。然而，我们认为最终还是由主 CDN 来负责确保 QoS 指标的满足，这时终端用户从某个内容提供商的网站上请求内容，而内容服务可以由对等关系中的任何一个 CDN 来提供。一个对等关系中有一个中心分派器（或某个授权 CDN），它一般由对等的发起者运行和管理，负责将请求重定向到多个对等成员处。多 CDN 之间的协议与内容提供商和主 CDN 之间的协议是彼此独立的，因此主 CDN 需要负责监控它所利用的任何对等 CDN 的性能，以便能履行对内容提供商的义务。

16.5.2　基于中介的对等 CDN

图 16.6 中显示了我们提出的两个模型中的第一个，用来协助对等 CDN 的创建。在这个例子中，“协作”内容分发是由内容提供商实现的，它们利用多个 CDN 的服务来确保覆盖适当的地理范围并同时满足性能指标。内容提供商有责任进行高效的内容分发。用户和内容提供商的交互流程为：①用户通过在网络浏览器中输入网址向内容提供商请求内容，用户请求直接传送到内容提供商的源服务器；②内容提供商使用自己的中介系统来选择向终端用户分发内容的 CDN，一个内容提供商能够选择多个 CDN（基于每个 CDN 的 QoS 性能、能力、当前负载以及所处地理位置等指标）向用户分发内容，被选择的 CDN 之间并不需要知道它们彼此是并行工作的，因为责任的管理和分离是由内容提供商来解决的；③达成一个内容提供商和 CDN 之间的基于策略的合作协议；④一旦建立了对等关系，被选择的 CDN 便使用自己的专有算法，选择最佳的 Web 服务器分发用户所请求的内容。

根据上述模型，为了加入到对等组合中，CDN 提供商可以通过彼此竞争来提升性能。内容提供商会持续监控 CDN 的性能。因此，内容提供商可以基于某 CDN 在以往相似内容上的分发情况来选择 CDN，也可根据某些策略向用户提供一定的优惠，这些策略既可以是简单的“付多少费给多少服务”，也可以是其他更复杂的策略。

16.5.3　由 QoS 驱动（定制）的基于中介的对等 CDN

上面的模型考虑到了每个可能参与创建对等 CDN 的成员性能，但并没有专门考虑终端用户所需要的 QoS 指标。用户可能因某些特殊情况（如瞬时拥塞）而具有动

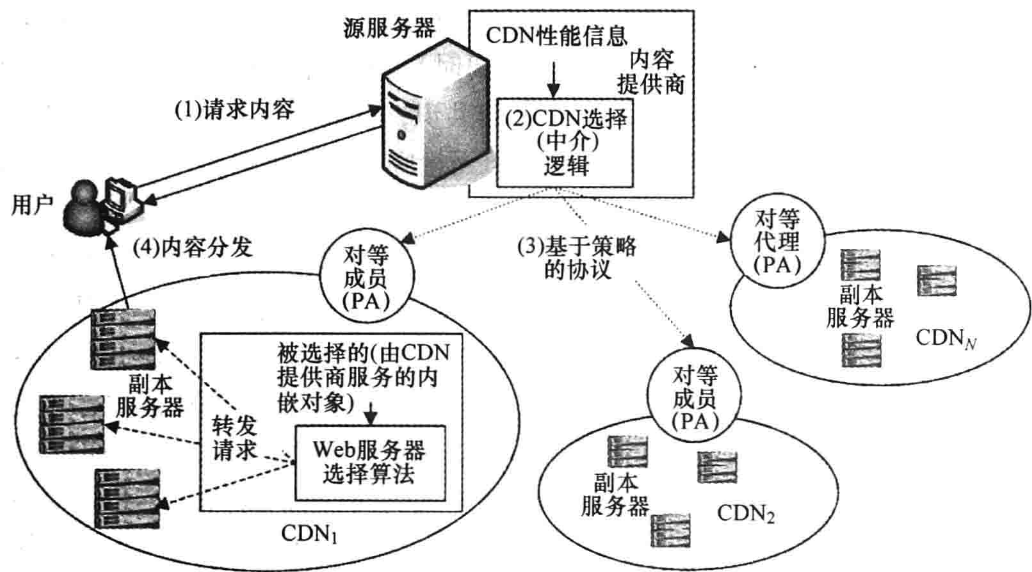


图 16.6 基于中介的对等 CDN 构建方法

态的需求，这就导致了“定制”的内容分发。因此，模型应考虑到用户自定义 QoS 的复杂情况，这取决于访问服务的用户类型。因此，在图 16.7 中我们展示了上述模型的改进版本，以协助对等 CDN 的形成。在这个改进版的模型中，内容提供商根据每个用户（或者用户群）的 QoS 需求来对参与者进行动态选择，其交互流程为：①用户根据特定的 QoS 需求从内容提供商处请求内容，请求到达内容提供商的源服务器；②内容提供商使用某个动态算法（基于用户自定义的 QoS）来选择 CDN；③内容提供商与它使用的（一个或多个）CDN 之间建立动态协议来确保满足用户的 QoS 指标；④一旦与所选 CDN 建立了对等，系统就从对等成员中的最优 Web 服务器处分发用户所请求的内容。

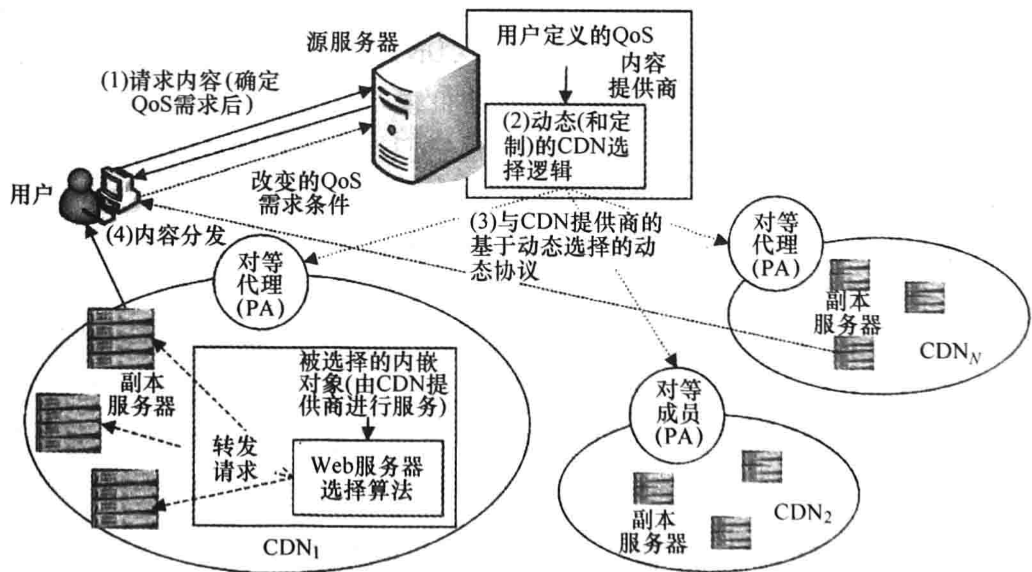


图 16.7 由 QoS 驱动（定制的）的基于中介的对等 CDN 构建方法

这样的对等组合是针对特定用户的，并且会随着 QoS 指标、范围、大小和能力的

变化而变化。内容提供商有责任通过动态对等组合来进行高效的内容分发。因此，如果某个对等组合在分发内容时不能满足用户的 QoS 指标，那么内容提供商就会和 CDN 提供商重新进行协商来建立新的对等组合。图 16.7 中显示了在最初的对等组合中，CDN_1 负责向用户分发内容。由于用户 QoS 需求的变化（由图中虚线显示），内容提供商需要相应地废除当前的 CDN 选择逻辑，重新建立一个新的对等组合。在新的对等组合中，CDN_N 是一个新的参与者，它使用自己的 Web 服务器向用户分发内容。

16.6　实现 CDN 对等时面临的挑战

来自技术和非技术层面的挑战（如商业和法律上的）阻碍了 CDN 对等的快速发展，所以必须解决这些问题。本节简单列举了 CDN 对等面临的常见问题。

（1）法律/版权问题

被分发的内容通常会涉及复杂的法律问题（如禁止发布或受版权保护的内容），这会阻碍 CDN 之间的自由合作。当对等伙伴之间涉及内容传输时，这种内容的交互必须考虑到与该内容相关的任何法律问题。例如，如果一个内容提供商需要的软件或者文件包含了某些政府禁止的逻辑和内容（即被限制访问），那么所有对等关系中的 CDN 提供商都要确保内容分发符合相关法律的规定。目前，CoDeeN 和 Coral 等学术 CDN 都对提供商分发的实际内容几乎没有控制，这样看来，这些对等系统中的参与者很可能会无意中违反法律规定。受版权保护的内容（如出版物和数字媒体等）特别需要仔细管理，以确保版权所有者的权益受到尊重和保护。某些 CDN 的操作（如缓存和复制）是由用户驱动而不是由内容提供商发起，而内容提供商更倾向于根据自己的意愿发布内容，而不是在未经它们同意的情况下将其内容随处缓存。

（2）遍布全球

正如前面章节所提到的，一般来说，CDN 提供商集中地管理其全球分布的设施。Akamai 和 Mirror Image 都拥有它自己的全球性网络，这能够满足其绝大多数用户的需求。的确，极高的覆盖面是这种大公司的竞争优势所在，并且使它们能够定位到高端客户市场。但是，很少有提供商能够与这些巨头公司的全球影响力相比，同时，这些大公司也缺乏与其他小型提供商建立对等的商业或运营兴趣。

（3）CDN 市场的整合

如果小型 CDN 提供商希望同大提供商在覆盖范围和性能方面进行竞争，直接对等可能会给小提供商带来优势。最近几年 CDN 市场中有很多并购的情况，其结果是从 20 ~ 30 个提供商减少到了 5 ~ 10 个大型提供商。很明显，较小的提供商会发现，与 Akamai 与 Mirror Image 等公司在覆盖范围和性能方面进行竞争非常困难，因此其后果往往是停业或者被大型 CDN 提供商收购。

（4）基于中介的 CDN 对等的挑战

由内容提供商自己选择多家 CDN 进行内容分发的方法无疑很诱人，尤其是当这些 CDN 提供商拥有丰富的资源和管理技术时。CDN 提供商可能会因为它们各自的优点而被选中（如位置、性能、价格等），并且他们通过联合可以提供给客户更好的性能体验。然而，当同时涉及多个提供商时，为确保终端用户拥有好的性能体验而执行

QoS 标准（实质上是试图创建一个鲁棒和可控的覆盖网络）可能会面临各种各样的挑战，尤其是当这些 CDN 实际上并没有协作关系，而只是简单地进行并行化运转时。

（5）基于 P2P 的 CDN 对等的挑战

最近十年，利用用户的带宽，以 P2P 方式协作分发内容变得越来越流行。尽管起初这是违背内容提供商意愿的（如 Napster、Gnutella 等），但最终内容提供商还是接受了 P2P 技术，特别是 BitTorrent。这主要是为了分发大量的可扩展性内容，其性能要远超过传统的分布式 CDN。内容提供商已经有效地利用 BitTorrent 来发布数字媒体（电影和音乐）、操作系统（如 Linux）和操作系统补丁、游戏和游戏补丁等。随着高速带宽的普及，终端用户带宽也得到了提高，用户在下载的同时上传一部分数据，内容提供商可以借此对用户进行平衡。然而，这种方法只对那些访问量很大的文件有效，而且由于某些文件并不会被足够多的用户制作“种子”，因而会导致较差的用户体验。因此，当发布内容的节点简单地是终端用户时，内容提供商很难保证任何特定的 QoS 指标下限，因为一旦终端用户接收完数据内容，它们可能就没有什么继续合作下去的动力了。

（6）缺乏对协作的激励机制

来自于互联网服务提供商（ISP）的强烈抵制，使得对等方法的前途更加不确定。ISP 并不希望内容提供商将内容分发的负担和成本转移到终端用户身上，因为最终还是会落到 ISP 自己头上。针对这一状况，许多 ISP 正积极阻止和限制 BitTorrent 和其他 P2P 流量，通过抑制其扩张来避免自身收入的缩水，以及减少 ISP 由此产生的配置额外设施的成本。许多在地理位置分布上比较孤立的国家（在所谓的边缘上），如澳大利亚和新西兰，其 ISP 属于特殊的情况，它们只能依赖少量的昂贵连接与北美和欧洲的网络互联。因此，这些地区的 ISP 所提供的宽带接入有固定的数据额度（而不是无限的），终端用户是受限的，这保证了 ISP 仍然有利可图。这些情况进一步阻碍了终端用户参与内容分发的协作，同时也不利于在内容发布中广泛采用协作的方式。

16.7 对等 CDN 的技术问题

对等 CDN 的部署存在独特的研究课题。在这一节中，介绍部署对等 CDN 将要面临的一系列问题。虽然针对 CDN 领域的相关问题存在一些解决方案，但 CDN 网络互联/对等的概念带来了额外的挑战。因此，为了给出下一步的研究方向，重点介绍与实现 CDN 对等有关的关键问题。

1. 对等 CDN 的负载分配

对等 CDN 的负载分配策略包括请求分配与重定向、负载传播（load dissemination）以及内容复制。而如何协调这些核心内容是负载分配策略成功与否的另一个重要考虑因素。

对等中的 Web 服务器在地理上分布于各处，为了将请求重定向和分配到这些服务器，负载分配策略除了需要考虑任务的大小（如内容请求的处理要求）之外，还需要考虑终端用户的位置、服务器负载以及终端用户和服务器之间的连接利用率。不

仅如此，它还要能够应对动态变化的情况，如瞬时拥塞以及其他不可预测的事件。除此之外，请求的分配和重定向还要能够运行于 CDN 的多个层面上，如 DNS、本地服务器集群的网关以及集群内的服务器之间[7,8]。商业 CDN 主要依靠在 DNS 层面使用一个基本的请求分配策略（如加权轮转或最小负载优先）来实现对终端用户的分配，以便通过更新 DNS 的记录来指向最适合的副本服务器[10]。在对等 CDN 中，可以通过 DNS 来分配终端用户（通过参与 CDN 的对等代理来定期更新它们的 DNS 记录），以及在恰当的时候通过 CDN 网关（也就是作为一个单一实体的调解器、PA 和策略库）进行重定向。

为了处理负载传播问题，可以使用峰值负载对流量进行建模，因为在这种情况下服务器的负载能力会面临最严峻的考验。负载信息的测量和传播可以在单个 CDN 和其他的 CDN 之间进行。一个负载指标可以用于测量计算机系统中单一资源的使用量。或者，它可以对多类资源进行综合测量，如 CPU 负载、内存使用率、磁盘分页和活动的进程等。这些负载信息需要在所有参与的 CDN 中及时、高效地传播，以最大限度地发挥它的作用。这些指标可以用来识别一些重要的情况，如正确建立 CDN 对等组合的条件（如当服务器或整个 CDN 网络过载）或判断何时 CDN 的资源利用率不足而可以将之提供给其他 CDN 提供商使用。对此，可以设想一种分层化的方法，CDN 中 Web 服务器的当前网络带宽和资源使用情况会以定期或基于阈值的方式向 CDN 网关报告，然后各相关 CDN 网关各自汇总其负载信息，而这些负载信息用于描述其组内服务器成员的负载状况。

在 CDN 内部，内容复制在源服务器及其他服务器中进行。现有 CDN 提供商（如 Akamai、Mirror Image 等）使用非协作的基于推送的方式，其中请求直接（通过 DNS）定向到最近的副本服务器[10]。如果请求的文件没有存储在那里，则副本服务器从源服务器获取内容。学术界已经提出了协作的基于推送的技术，它使用一个贪婪全局启发式算法[6]将内容推送到参与的 CDN 镜像服务器中。在这种方法中，请求被直接定向到最近的镜像服务器，或者如果附近没有合适的镜像服务器，请求就被定向到源服务器。在对等 CDN 中，这种复制手段延伸到参与其中的其他 CDN 服务器，促进了服务器之间可用资源的共享。

总之，对等 CDN 之间分配负载需要解决以下问题：

1）如何产生一个动态请求分配和重定向策略用于计算系统运行过程中请求路由的理想参数？

2）如何才能确保减少服务器负载、减少带宽消耗（由特定的 CDN 服务器消耗）以及提高内容分发的性能？

3）参与的 CDN 如何通过协作来复制内容，以便为所有的参与各方提供满意的解决方案？

4）采取什么措施来确保缓存对象不过期？如何处理不能缓存的对象？

2. CDN 的协调

上述负载分配的核心技术问题的任何解决方案都必须在对等组合的所有参与者之间进行协调，这样才能提供高性能和用户满意的 QoS。因此，必须开发一种用于协作

的中间件来确保解决方案能正确执行，以解决这些核心问题。为此，需要解决的主要问题是：需要采取哪种协调机制才能确保有效性，并且使得对等 CDN 能够发展和可扩展？

3. 服务和策略的管理

在对等 CDN 中，应该根据用户的偏好进行内容管理。因此，一个管理分布式内容的综合模型必须要考虑到用户的偏好。为了解决这个问题，可以对内容进行个性化处理来满足特定用户（或者用户群）的喜好。例如，在一些个性化的网站中，使用数据挖掘技术从内容请求和相关数据中自动获得用户的兴趣所在[14]。在 CDN 中，可以通过对流量、定价以及记账等事务的正确管理，利用数据挖掘技术使性能得到显著的提高。在这方面，需要解决以下问题：

1）如何在用户可以访问的设施服务中添加增值服务？

2）参与各方需要协商什么类型的 SLA？要生成什么策略来支持 SLA 协商？

3）如何能够及时地启动自主策略协商来支持一个具有时间敏感性的对等组合？

4. CDN 中内容和服务定价

为了实现对等 CDN 参与者之间持续分享资源，必须确保所有参与各方都有足够的激励存在，这就需要配置正确的价格、付费和管理系统。这一方面要解决的主要问题是：

1）在价值表达（内容和服务需求的表达和估值）、价值转换（将需求转移到内容和服务分发中）以及价值实施（不同内容和服务的选择和分布的实施机制）这几个方面需要使用什么机制？

2）在竞争环境中 CDN 提供商如何在保持供求平衡的同时达到收益最大化？

16.8 结论

目前内容网络和内容联网技术的发展趋势已经引起了人们对 CDN 网络互联的兴趣。找到不同 CDN 与其他内容网络之间协调、合作的方式，对于实现更好、更全面的服务至关重要。在本章中，我们提出了 CDN 网络互联的一种方法，它致力于在 CDN 的服务需求和由于部署特定客户所带来的高运营成本以及因此造成的资源过度配置等情况之间做出平衡。在我们所提出的方法中，CDN 的可扩展性和 CDN 之间的资源共享可以通过对等技术得到性能提升，这就为 CDN 的发展开辟了全新的视角。这一章中，我们还提出了两个新的模型来支持 CDN 对等技术，并指出了相关的研究挑战。实现 CDN 对等的概念，将成为对发展中的内容网络的一个重要贡献。

致谢

本章使用的一些材料最初出现在 IEEE DSOnline[5]、UPGRADE - CN'07[17]，以及 TCSS Doctoral Symposium - CCGrid'07[18]中。部分研究得到了 Australian Research Council（ARC）的 discovery project 基金、Department of Education，Science，and Training（DEST）的 International Science Linkage（ISL）基金的支持。本章部分内容的产生也极大地得益于与 K. H. Kim 和 Kris Bubendorfer 的讨论。

参 考 文 献

[1] Amini, L., Shaikh, A., and Schulzrinne, H. Effective peering for multi-provider content delivery services. In *Proc. of 23rd Annual IEEE Conference on Computer Communications (INFOCOM'04)*, pp. 850–861, 2004.

[2] Arlitt, M. and Jin, T. Workload characterization of the 1998 world Cup Web site. *IEEE Network*, 14:30–37, 2000.

[3] Assuncao, M., Buyya, R., and Venugopal, S. Intergrid: A case for internetworking islands of grids, *Concurrency and Computation: Practice and Experience (CCPE)*, Willey press, New York, USA, 2007.

[4] Biliris, A., Cranor, C., Douglis, F., Rabinovich, M., Sibal, S., Spatscheck, O., and Sturm, W. CDN brokering. *Computer Communications*, 25(4), pp. 393–402, 2002.

[5] Buyya, R., Pathan, M., Broberg, J., and Tari, Z. A case for peering of content delivery networks, *IEEE Distributed Systems Online*, 7(10), 2006.

[6] Cardellini, V., Colajanni, M., and Yu, P. S. Efficient state estimators for load control policies in scalable Web server clusters. In *Proc. of the 22nd Annual International Computer Software and Applications Conference*, 1998.

[7] Cardellini, V., Colajanni, M., and Yu, P. S. Request redirection algorithms for distributed Web systems. *IEEE Trans. on Parallel and Distributed Systems*, 14(4), 2003.

[8] Colajanni, M., Yu, P. S., and Dias, D. M. Analysis of task assignment policies in scalable distributed Web-server systems. *IEEE Trans. on Parallel and Distributed Systems*, 9(6), 1998.

[9] Day, M., Cain, B., Tomlinson, G., and Rzewski, P. A Model for Content Internetworking. IETF RFC 3466, 2003.

[10] Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman R., and Weihl, B. Globally distributed content delivery. *IEEE Internet Computing*, pp. 50–58, 2002.

[11] Freedman, M. J., Freudenthal, E., and Mazières, D. Democratizing content publication with coral. In *Proc. of 1st Symposium on Networked Systems Design and Implementation*, San Francisco, CA, pp. 239–252, 2004.

[12] Guo, L., Chen, S., Xiao, Z., and Zhang, X. Analysis of multimedia workloads with implications for internet streaming. In *Proc. 14th international Conference on World Wide Web (WWW)*, pp. 519–528, 2005.

[13] Iyengar, A. K., Squillante, M. S., and Zhang, L. Analysis and characterization of large-scale Web server access patterns and performance. *World Wide Web*, 2(1–2), 1999.

[14] Mobasher, B., Cooley, R., and Srivastava, J. Automatic personalization based on Web usage mining, *Communications of the ACM*, 43(8), pp. 142–151, 2000.

[15] Padmanabhan, V. N. and Sripanidkulchai, K. The Case for Cooperative Networking. In *Proc. of International Peer-To-Peer Workshop (IPTPS02)*, 2002.

[16] Pai, V. S., Wang, L., Park, K. S., Pang, R., and Peterson, L. The dark side of the Web: an open proxy's view. In *Proc. of the Second Workshop on Hot Topics in Networking (HotNets-II)*, Cambridge, MA, USA, 2003.

[17] Pathan, M., Broberg, J., Bubendorfer, K., Kim, K. H., and Buyya, R. An architecture for virtual organization (VO)-based effective peering of content delivery networks, UPGRADE-CN'07, In *Proc. of the 16th IEEE International Symposium on High Performance Distributed Computing (HPDC 2007)*, Monterey, California, USA, 2007.

[18] Pathan, M. and Buyya, R. Economy-based content replication for peering CDNs. TCSC Doctoral Symposium, In *Proc. of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007)*, Brazil, 2007.

[19] Pierre, G. and van Steen, M. Globule: A platform for self-replicating Web documents. In *Proc. of the 6th International Conference on Protocols for Multimedia Systems (PROMS'01)*, The Netherlands, pp. 1–11, 2001.

[20] Pierre, G. and van Steen, M. Globule: a collaborative content delivery network. *IEEE Communications*, 44(8), 2006.

[21] Turrini, E. An architecture for content distribution internetworking. Technical Report UBLCS-2004-2, University of Bologna, Italy, 2004.

[22] Verma, D.C., Calo, S., and Amiri, K. Policy-based management of content distribution networks, *IEEE Network*, 16(2), pp. 34–39, 2002.

[23] Wang, L., Park, K. S., Pang, R., Pai, V. S., and Peterson, L. Reliability and security in the CoDeeN content distribution network. In *Proc. of Usenix Annual Technical Conference*, Boston, MA, 2004.

[24] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S, Huynh, A., Carlson, M., Perry, J., and Waldbusser, S. Terminology for policy-based management, IETF RFC 3198, 2001.

[25] Zhao, W. and Schulzrinne, H. DotSlash: A self-configuring and scalable rescue system for handling Web hotspots effectively. In *Proc. of the International Workshop on Web Caching and Content Distribution (WCW)*, Beijing, China, 2004.

国际视野　科技前沿

国际信息工程先进技术译丛

《内容分发网络》
《Android系统安全与攻防》
《移动云计算：无线、移动及社交网络中分布式资源的开发利用》
《认知视角下的无线传感器网络》
《计算机网络仿真OPNET实用指南》
《移动无线信道》（原书第2版）
《LTE-Advanced：面向IMT-Advanced的3GPP解决方案》
《声学成像技术及工程应用》
《认知无线电通信与组网：原理与应用》
《LTE/SAE网络部署实用指南》
《网络性能分析原理与应用》
《云连接与嵌入式传感系统》
《IP地址管理原理与实践》
《自组织网络：GSM，UMTS和LTE的自规划、自优化和自愈合》
《实现吉比特传输的60GHz无线通信技术》
《LTE自组织网络（SON）：高效的网络管理自动化》
《UMTS中的LTE：向LTE-Advanced演进》（原书第2版）
《无线传感器及执行器网络》
《UMTS中的WCDMA－HSPA演进及LTE》（原书第5版）
《认知无线电网络》
《网络融合——服务、应用、传输和运营支撑》
《UMTS中的LTE:基于OFDMA和SC-FDMA的无线接入》
《高性能微处理器电路设计》
《吉规模集成电路互连工艺及设计》
《高级电子封装》（原书第2版）
《基于4G系统的移动服务技术》
《移动无线传感器网——技术、应用和发展方向》
《UMTS蜂窝系统的Q0S与Q0E管理》
《UMTS-HSDPA系统的TCP性能》
《基于射频工程的UMTS空中接口设计与网络运行》
《未来UMTS的体系结构与业务平台：全IP的3GCDMA网络》
《环境网络：支持下一代无线业务的多域协同网络》
《基于蜂窝系统的IMS—融合电信领域的V0IP演进》
《蜂窝网络高级规划与优化 2G/2.5G/3G/——向4G的演进》
《微电子技术原理、设计与应用》
《多电压CMOS电路设计》
《P2P系统及其应用》
《IPTV与网络视频：拓展广播电视的应用范围》
《下一代无线系统与网络》



上架指导　工业技术／电子信息／通信技术

ISBN 978-7-111-40052-3　定价：98.00元